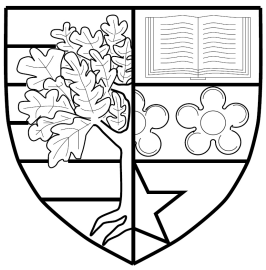
INCREASING THE ROBUSTNESS OF AUTONOMOUS SYSTEMS TO HARDWARE DEGRADATION USING MACHINE LEARNING

by

Georgios Fagogenis



Submitted for the degree of
Doctor of Philosophy

Institute of Sensors, Signals and Systems
School of Engineering and Physical Sciences
Heriot-Watt University

June 2016

I would like to dedicate this thesis to my family

Abstract

Autonomous systems perform predetermined tasks (missions) with minimum supervision. In most applications, the state of the world changes with time. Sensors are employed to measure part or whole of the world's state. However, sensors often fail amidst operation; feeding as such decision-making with wrong information about the world. Moreover, hardware degradation may alter dynamic behaviour, and subsequently the capabilities, of an autonomous system; rendering the original mission infeasible.

This thesis applies machine learning to yield powerful and robust tools that can facilitate autonomy in modern systems. Incremental kernel regression is used for dynamic modelling. Algorithms of this sort are easy to train and are highly adaptive. Adaptivity allows for model adjustments, whenever the environment of operation changes. Bayesian reasoning provides a rigorous framework for addressing uncertainty. Moreover, using Bayesian Networks, complex inference regarding hardware degradation can be answered.

Specifically, adaptive modelling is combined with Bayesian reasoning to yield recursive estimation algorithms that are robust to sensor failures. Two solutions are presented by extending existing recursive estimation algorithms from the robotics literature. The algorithms are deployed on an underwater vehicle and the performance is assessed in real-world experiments. A comparison against standard filters is also provided.

Next, the previous algorithms are extended to consider sensor and actuator failures jointly. An algorithm that can detect thruster failures in an Autonomous Underwater Vehicle has been developed. Moreover, the algorithm adapts the dynamic model online to compensate for the detected fault. The performance of this algorithm was also tested in a real-world application.

One step further than hardware fault detection, prognostics predict how much longer can a particular hardware component operate normally. Ubiquitous sensors in modern systems render data-driven prognostics a viable solution. However, training is based on skewed datasets; datasets where the samples from the faulty region of operation are much fewer than the ones from the healthy region of operation. This thesis presents a prognostic algorithm that tackles the problem of imbalanced (skewed) datasets.

Acknowledgements

I would like to offer my many thanks to Prof. David Lane and Dr. David Flynn, my research supervisors for their guidance and support throughout these three years of study at Heriot-Watt University. Notably, Prof. Lane has shown great confidence in my research and provided a comfortable working environment. Moreover, Prof. Lane funded several trips to international conferences, helping me in this way to stay up-to-date and get exposed to high-end research.

I would also like to thank Dr. Zeyn Saigol for his scientific advice, as well as for his patience while proofreading my manuscripts. I am particularly grateful to Dr. Tom Larkworthy for providing me with various research stimuli in the first one year and a half, during which we shared the same office.

Special thanks to all my fellow Ph.D. students and Research Associates within the Ocean Systems Lab for the research discussions, the technical tips and in particular Mr. Valerio De Carolis for providing me with experimental data from the field. I wish to thank Mr. Len McLean, our lab technician. Even though, we never worked together he largely contributed to having a functional research platform in the lab.

At this point, I wish to thank my former advisers, namely Prof. Kostas Kyriakopoulos, Prof. Bradley Nelson, Prof. Oliver Brock and Dr. Christos Bergeles. Each one of them contributed to shaping a strong research working profile, which eventually led to the completion of this thesis.

I would like to express my very great appreciation to my loving parents and sister for their unlimited and multifaceted support over the complete course of my studies. I consider my professional accomplishments as jointly achieved by my family and me. Both my parents believed in my skills from the very beginning of my school years and gently kept me on track when necessary.

Many thanks deserve to my friends for putting up with my mood swings and for always calming me down when I most needed it. Everything would have been much harder without my better half Eirini. Therefore, I would like to thank Eirini Papadaki separately for her endless patience, love and support throughout these years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

An *autonomous* system performs a prescribed task without any external supervision or support. Autonomous systems (AS) have long been used for tasks where human intervention is problematic. Space exploration and unmanned -aerial and underwater- missions are among the most common applications of autonomous systems. Also, many researchers investigate the use of such systems in search-and-rescue operations. Apart from scientific missions and research, autonomous systems are also ubiquitous in industry. Over the last decades, the autonomy of production has increased significantly. A typical example from the energy sector is that of 'smart grids'. The smart grid adapts autonomously to shifts in client demand and changes in the network topology. In this way, the network efficiency is optimised based on real-time information about clients' power consumption. In general, there is an observable increase in systems that are intelligent to carry out a task and adapt to changes autonomously.

Making a system fully autonomous is very difficult. Apart from physical constraints, (e.g., energy autonomy, data storage and communications) autonomous agents require sophisticated decision mechanisms to plan their actions. System complexity, as well as the inherent unpredictability of the operating environment, further exacerbate this problem. For this reason, systems are often designed to be autonomous in narrow regions of their operating range. Control theory provides

the means to achieve system stability in a neighbourhood around an operational point. Optimal control schemes drive the system along trajectories that optimise certain criteria. Robust control techniques assist a system's function in the presence of bounded disturbances.

Global autonomy (i.e., not within a specific region of operation) requires more sophisticated deliberation capabilities. Autonomous systems sense and reason about the environment; appropriately adjusting their actions towards fulfilling the mission at hand. Failure to complete a mission often comes as a result of the uncertainty about the world; this includes uncertainty about the system's integrity as well. Hardware failures, for example, may render a system inoperable, and hence incapable of achieving its goal. In such cases, human intervention is needed to restore the lost functionality.

To this end, the notion of reliability, namely preserving the functionality of the system, is an area of significant importance in the development of truly autonomous systems. Using information about the health state of the system helps to plan in a way, that the effect of hardware failures on the mission's outcome is kept to a minimum. This is done by either preventing failure or by treating failures in an optimal fashion. An autonomous underwater agent, for example, may deliberately use just part of its actual locomotion capabilities (e.g., move with reduced speed), to protect a thruster that is about to fail.

Several issues need to be addressed on the way towards persistent autonomy. Highly non-linear dynamics often govern real-world autonomous systems. An accurate dynamic model will play an instrumental role in the development of the presented algorithms. It is important to have a convenient and accurate method for dynamic modelling. Moreover, variability in either the operational conditions or the dynamic parameters requires a flexible representation; one that can easily adapt to dynamic alterations.

An integral part of practically every modern engineering asset is sensor instrumentation. Sensors are widely used to measure quantities relevant to a system. The information of the sensors is often used to make decisions at runtime. Quite

commonly, however, sensors fail; leaving the system with unreliable information, if any at all. Long term autonomy requires that the system detects whether there is a problem with the sensors, and consequently it relies on alternative sources of information to operate further.

Apart from dynamic and operational variability, autonomous systems may exhibit altered dynamic behaviour due to a hardware failure. An underwater vehicle, for example, may lose acceleration capabilities because of a defective thruster. Even worse in some cases, the vehicle might lose a full degree of freedom; i.e., the vehicle will not be able to move along one dimension that it previously could. Identifying such a situation provides the opportunity for reaction; i.e., that the policy of the autonomous system is revisited to accommodate the new situation. In this way, alternative capabilities may be exploited to pursue the achievement of the mission persistently.

Assuming that a sensor failure is more probable than a dynamic alteration (and vice versa) simplifies the problem significantly. It is hard to understand whether the autonomous system started behaving differently, or if a sensor has failed and the discrepancy originates from the observations. To properly address this issue, additional information may be required.

Whereas fault mitigation systems for sensor failures and model alterations consider what has happened, prognostics pertain to the computation of how long a component will be functional. This information will be taken into account during mission planning; producing as such more reliable plans that maximise the probability of mission completion. Prognostics can also alter the agent's policy in real-time. Given that the life expectancy of a component falls critically low, the agent may switch to an alternative plan. For example, the power grid may change topology (e.g., disconnect/connect an extra generator) after realising that one certain hardware component is about to fail. An underwater vehicle may drop the inspection of remote targets - despite a hypothetical higher utility - and visit targets that are nearby, in light of an impending thruster failure.

1.2 Approach

As mentioned earlier, modern systems are equipped with a plethora of diverse sensors. The sensors provide useful information about the monitored system. This information is often high-dimensional; hence, it requires suitable algorithms to extract the part that is relevant to the task. Contemporary data mining and machine learning algorithms constitute a powerful arsenal for this matter.

As previously mentioned, the autonomous agent needs to infer about the state of the world. Such inference is subject to all sorts of uncertainty. For this reason, a suitable framework, i.e., one that considers uncertainty, is required. This framework is provided by Bayesian Networks (BN). A Bayesian Network is a graphical representation of conditional dependencies between a set of random variables. In other words, a BN is used to decompose the joint probability of a set of random variables (i.e., variables that are sampled from a probability distribution) to a product of simpler factors; exploiting the conditional dependences of the latter. Given the joint probability distribution, any query combination can be answered by setting the variables that represent the observations and marginalising (summing over all possible events) the variables that are neither evidence nor query variables. Marginalisation is what hinders efficient computation of the queries; sometimes making them even computationally intractable. To this end, several, algorithms have been developed both for answering the exact question or an approximate one; i.e., *exact* and *approximate* inference respectively.

The choice between exact and approximate inference depends on the constraints of the application. Exact inference accurately answers a query, whereas approximate inference provides a mere estimation. However, approximate inference is, in general, much faster. In what follows, Bayesian Networks are widely used to answer queries about the system's hardware integrity. Due to real-time constraints, approximate inference was the machinery employed in this thesis. Non-linear regression and Bayesian inference (together with some other machine learning algorithms) have been combined to advance the state-of-the-art in fault detection and mitigation. Bayesian reasoning has met great success and appreciation within most research

fields. It is a straightforward way of explaining the world in a probabilistic fashion. Prior knowledge is rigorously incorporated in the initialisation of the probability distribution. Accumulated evidence, in turn, alters the initial distributions to abide with the observation in the world. With the use of probability distributions, uncertainty is addressed in an elegant and efficient manner.

Machine learning offers several options for approximating non-linear relations. Such algorithms are presented with a set of examples (training data). Given adequate training, the algorithm can compute estimates for new queries. Specifically, non-parametric (kernel-based) algorithms have been used in this thesis. The main reason for that is that kernel methods do not require any assumptions; in particular, about the class of candidate functions. Moreover, machine learning provides several algorithms that can adapt at runtime. In this way, the problem of dynamic variability can be alleviated by using a core model, and use the adaptive traits of the algorithm for fine-tuning.

This technology has matured within the cycles of computer science. Nevertheless, other engineering fields have not fully exploited the power of machine learning. In particular, fault mitigation and prognostics can benefit strongly from such algorithms. Robotics, and, in particular, autonomous underwater vehicles, are more up to date; albeit the current state-of-the-art lags significantly from computer science research frontier. The goal of this thesis is to tailor cutting-edge algorithmic technology to the engineering specifications of autonomous systems. The following chapters attempt to bridge this gap by using machine learning techniques to develop robust algorithms to be used in real-world autonomous systems.

1.3 Contribution

The contribution of this thesis is that it bridges the gap between the algorithmic advances in computer science and the real-world demands of modern autonomous systems. A versatile reasoning system has been developed that can deal with sensor failures in real time. The latter was extended appropriately, to reason for alterations in the dynamics. This capability has been showcased again on underwater

navigation; particularly, for thruster failure detection. Adaptive modelling, and in particular Locally Weighted Projection Regression (LWPR), have been widely explored. The integration of LWPR within a Bayesian Filtering framework is another contribution of this work. Even though the algorithm was demonstrated for underwater navigation, it provides a general substrate for sensor failure prone systems with highly non-linear dynamics. The algorithms presented in the following chapters require minimum tuning by the user. Expert knowledge is incorporated formally by the use of prior distributions. A novel algorithm for data-driven prognostics has been developed. The presented algorithm not only exhibits adaptive behaviour to remedy ageing and dynamic variation; it also, through the use of a robust high-end classifier, tackles the problem of imbalanced datasets, which often appears in the prognostics literature. The main points of contribution accompanied by the respective publication, if available, are summarised below:

- the thesis showcases the application of adaptive, data-based modelling to underwater dynamic modelling for navigation [156]
- we combined LWPR with bayesian filtering to yield a navigation algorithm that is robust to common sensor failures and tested the algorithm on experimental data [184]
- we used mixtures of Gaussians and bayesian filtering to yield an algorithm that can distinguish sensor failures from dynamic alterations; the algorithm was tested on experimental data [185]
- the thesis presents a new data-driven prognostic algorithm that solves the problem of misclassification of failures due to misrepresentation of the latter in the training dataset [186]
- the thesis provides a systematic way of designing and implementing a bayesian filter from scratch; theory about approximate inference is presented as the means to derive the recursive equations of any bayesian filter.

1.4 Structure

The structure of this thesis is as follows: Chapter 2 provides an in-depth review of the state of the art in fault detection and prognostics. Moreover, a separate section in that particular chapter is devoted to underwater navigation. This is mainly because underwater navigation was the main application scenario, on which most of the algorithms have been showcased. Chapter 3 introduces general concepts in dynamic modelling. Moreover, chapter 3 explains the particular algorithm that has been used for dynamic modelling. Next, chapter 4 describes two outlier robust algorithms that have been used to mitigate sensor failures. Chapter 5 extends the previous algorithms to the case of actuator failures. Following that, chapter 6 presents a novel algorithm for data-driven prognostics. Lastly, chapter 7 concludes this brief; summarising the main achievements and presenting possible future directions that would further advance the state-of-the-art in autonomous systems.

Chapter 2

Relevant Work

Many of the concepts, utilised in this thesis, originate from recursive estimation; in particular, *recursive bayesian estimation*. In recursive Bayesian estimation, the goal is to estimate the unknown state of a system given measurements and a model for the system's dynamics. Increasing hardware robustness, in this context, means to identify which source of information to trust more; i.e., sensor measurements, or estimation of the model dynamics. When a sensor fails, model predictions can help to continue operation or -at least- to stop operations safely. Model estimates, on the other hand, indicate whether the autonomous system operates as expected. Recursive estimation bears considerable resemblance to diagnostic algorithms since many of the latter are special cases of recursive state estimation problems. Nevertheless, diagnostics are more application dependent, whereas recursive estimation attacks the problem in a more general way.

Increasing the robustness of autonomous systems combines several fields of research. Mainly, for hardware degradation, several papers have reported on algorithms that treat faults at different levels. Some approaches deal with failures that have occurred in the present or the recent past. These methods fall into *Diagnostics*. Diagnostics is a well-established field of engineering, albeit it can significantly benefit from recent advances in Machine Learning. Briefly, diagnostics attempt to detect, isolate and recover from failure. Section 4.2 offers an extended review of the field.

Conversely, prognostics revolve around the notion of *Remaining Useful Life*;

that is, to estimate how long a hardware component can operate above a predefined threshold level. The above problem is, in fact, a diagnostic problem shifted in the future. Hence, prognostic algorithms often share many similarities with diagnostic algorithms. However, prognostics is still a budding field of research; it has not reached the level of maturity of diagnostics. That is because, involved systems are hard to model; hence, hard to predict robustly future behaviour. Section 2.2 describes some early attempts, as well as recent advances in prognostic algorithms.

Much of the experimental validation of this thesis is performed on an Autonomous Underwater Vehicle. Specifically, the developed algorithms were applied to aid navigation in case of hardware (sensor and actuator) failures. For this reason, the last section of this chapter 2.3 is devoted to work from the underwater navigation literature.

2.1 Diagnostics

Diagnostics pertain to the detection and isolation of both abrupt and incipient faults in machines. Abrupt faults occur suddenly, whereas incipient faults cause a gradual degradation before complete failure. Another categorisation is based on how a defect influences the process state. Additive faults (e.g. sensor and actuator faults) corrupt the state evolution by adding an unknown quantity; the fault is modelled as an external disturbance entering the system. On the other hand, parameter faults act in a multiplicative manner. Diagnostic systems detect deviations from normal operation as a result of faults. It is often hard to infer whether such a difference is due to a defect or caused by other factors (normal ageing, a shift in operating conditions). Also, for the sake of reliability, the number of false alarms needs to be kept to a minimum [2]. Following fault detection, the system attempts a *diagnosis*; it tries to reason about what led the process to an out-of-control situation.

Many diagnostic algorithms reason about failures by examining features of raw signals [3]. Depending on the application, features reveal different aspects that could potentially influence diagnostic performance. Signals may be processed in either the *time domain* or the *frequency* domain. Time domain analysis involves statistical

computation of signal properties in the signal's original form; i.e., as recorded by the sensors. Examples of time domain features include, a signal's mean and standard deviation, as well as signal peak and peak-to-peak intervals. Often, high-order statistics are also computed (e.g., root mean square, skewness, kurtosis). A prevalent amongst time-domain approaches is the Time Synchronous Average (TSA); where, before computing the above statistics, several time series are averaged to decrease the signal's noise.

An overview of TSA approaches may be found in [4]; moreover, [5] discusses the shortcomings briefly. Commonly, Autoregressive (AR) models have been applied to time-domain signals. For example, Poyhonen et al. [6] fitted an autoregressive model to vibration signals, originating from an electrical motor. Next, the coefficients of the autoregressive model have been utilised as features for diagnostic inference. Apart from standard autoregression models, several time-series approximation tools have been put to the task of feature extraction. Specifically, Baillie and Mathew [7] provide a comparison of AR models, with standard neural networks and radial basis function networks. Because of sensor abundance of modern autonomous systems, feature extraction can often become a high-dimensional problem. To this end, Garga et al. [8] combined autoregressive models with dimensionality reduction. Autoregression, expressed as a state-space model, has also been exploited for the analysis of vibration signals [9]. In general, time-domain approaches suffer from complexity in building the model, as well as, from determining the order of the model. For these reasons, many researches have shifted focus to different feature extraction regimes.

Contrary to time-domain analysis, an alternative approach is to process the signal after transforming it to the frequency domain. This is particularly useful in the case of periodic phenomena (e.g., in the diagnostics of rotating machinery). Moreover, often hardware degradation is accompanied by unexpected vibrations. The latter would appear as an unexpected peak in the respective frequency range. This undesired vibration may be easily isolated. The most common way to perform such a signal transformation is the Fourier Transform; particularly, the Fast Fourier

Transform (FFT). Most commercial engineering software provides a FFT as a built-in functionality.

The prevalent metric that appears in frequency domain methods is the *power spectrum* of the signal. The latter is defined as the expected value of the squared Fourier transform of the original signal, and is indicative of the presence of each signal frequency. Graphical tools for the facilitation of spectrum analysis have been widely used for fault detection and diagnostics; examples of the latter include frequency filters, envelope analysis [10], side band structure analysis [11], Hilbert transform [12, 13]. As mentioned before, power spectrum analysis has ubiquitous success stories in the diagnostics literature. However, other spectra-analysis methods also provide an interesting insight in certain cases. One class of approaches, namely *Cepstrum* relies on the detection of harmonics and side-band combinations of frequencies. A more rigorous definition of cepstrum can be found in [14]. Among several definitions, the most commonly used is as follows: cepstrum is the inverse Fourier transform of the logarithmic power spectrum. A slightly modified analysis is presented in [15]. Some approaches utilize high-order spectra for fault detection, especially in the case of non-Gaussian signals; i.e., when signals are corrupted with non-Gaussian noise. Specifically, *bispectrum* and *trispectrum*, namely the Fourier transform of the third and forth order statistics of the original time-domain waveform respectively, have been extensively used for diagnostics on mechanical systems. For example, bispectrum analysis have been applied by [16] for fault detection pertaining to gears; [17] applied bispectrum for bearing diagnostics. Bispectrum has been also used for rotating [18] and induction machines [19, 20]. Bispectrum and trispectrum as a means to bearing fault diagnostics was discussed in [21]. A new approach, namely holospectrum, was first reported by Qu et al. [22]. Holospectrum exploits all sort of signal traits, including phase, amplitude and signal frequency. Application of holospectrum to fault detection is reported in [23].

Frequency domain analysis is restricted to stationary signals. The stationarity assumption is often not viable for failing hardware. To remedy this limitation, time-frequency analysis examines the signal in both the time and frequency domain.

Specifically, time-frequency techniques use two-dimensional energy functions with respect to time and frequency. Emerging patterns of this energy function facilitate accurate diagnostics. Another way to tackle the problem of non-stationary waveforms is to approximate them locally as stationary; known in the literature as the Short-time Fourier transform (STFT) [24].

Most prominent on this analysis regime is the wavelet transform. In wavelet transform analysis the signal is decomposed into a series of periodic functions with different frequencies at different times; i.e., not merely sinusoid functions as in the Fourier transform. Similarly to the phase in the frequency domain a translation parameter shifts the wavelets to reconstruct the original signal. The main advantage in using wavelets is that they yield non-uniform resolution in different parts of the time-frequency domain. Specifically, for signals with high-duration low frequency content and short duration high frequency, the wavelet transform will have high frequency resolution in low frequencies and high time resolution at high frequencies. Moreover, the wavelet transform is robust to noise infected signals. [25] report the use of wavelet transform for the detection of faults in gears. [26], in turn, provide an example of the wavelet transform applied on gears; whereas, [27] utilize the same diagnostic principle on other mechanical systems. A comparison between the wavelet transform and other available vibration analyses is presented in Dalpiaz and Rivola [28]. The wavelet transform attracted the strong interest of many researchers in the field: Addison et al. [29] used low-frequency wavelets for feature detection. The Haar wavelet has been employed by [30]. In an effort to combine the advantages of both methods, [31] composed wavelet transform with standard Fourier transform to aid the extraction of features related to component failure. Another common approach for fault detection is known as *wavelet packet transform* [32]. Using the basis from the wavelet packet decomposition, the signal can be filtered and the more salient wavelets are used to compose features for fault detection. This approach is known as *basis pursuit* [33]. A detailed review on the applications of wavelet packet transform in the context of machine reliability can be found in [34].

Characterisation of faults in terms of signal features has had various success

stories. However, deciding which features to consider for fault detection is often a daunting process, for it requires extensive understanding of the dynamics of each particular failure mode. Moreover, the resulting features require additional sensors. In many systems, space limitation prohibits auxiliary instrumentation. Frequency methods have exhibited robustness to noise and have been successfully applied in versatile fault detection applications. Successful as it is in the component level, feature based diagnostics fails to scale to complex systems. Finally, feature based techniques provide detection and isolation, yet they do not assist continuation of operation; they merely provide information on what went wrong. Nevertheless, feature-based diagnostics constitutes a valuable tool for post-failure evaluation of the incurred hardware damage.

Quite often engineers install multiple components of the same functionality(hardware redundancy), to ensure the safe operation of the plant in case of failure. The same concept can be applied to diagnose defective sensors. By having multiple sensors measuring the same or similar signals, detection and isolation of faulty sensors becomes quite easy. Nevertheless, this approach is intrusive and often not applicable in practice. Furthermore, it increases the cost of instrumentation as well as the computational cost of information processing. For the case of autonomous systems, in particular, hardware redundancy is hardly ever the choice; unless, a mission critical component is proven to fail very frequently. Rather than hardware redundancy, most systems leverage either the concept of *analytical redundancy* - model-based approaches (see section 2.1.1)- or historical data (see sections 2.1.2, 2.1.3) to perform fault detection. There are three main approaches for fault detection in the literature:

- Model-based approaches
- Statistical approaches
- Artificial Intelligence approaches

In the remaining part of this section, the most representative methods for the aforementioned categories are presented.

2.1.1 Model-based approaches

Model based approaches use a mathematical representation to estimate the future state and output of the system. This estimation, combined with measurements from the sensors yields the residuals that are used for fault detection. Next, the system infers whether these residuals are indicative of faulty behaviour (residual evaluation). For this reason, the generated residuals must be zero - zero mean in practice - in the absence of a fault [35]. Some systems are also able to provide deeper insight on the nature of the failure based on the form of the residual.

In model-based approaches the residuals are generated using Analytical Redundancy Relations (ARR). An Analytical Redundancy Relation is a constraint between the observed variables of a system [36]. The combination of all these relations builds up a model of the system which can estimate the value of the output variables. This model combined with the readings from the sensors produces the residuals.

One early expression of analytical redundancy, on which several fault detection methods have been based, is that of *parity relations*. Complying to the ARR paradigm, parity relation based fault diagnostics relied on the residuals between the system and model output. Next, a linear transformation is applied to rotate the data along the dimension most discriminative for the faulty behaviour. The described two-step process constitutes a *residual generation* mechanism; indicative for one or a set of faults that are known *a priori*. The set of faults depends on prior knowledge about the fault dynamics. The linear transformation that is applied on the generated residuals is often expressed as a linear filter. Residual generation filters are built in a way that facilitates *fault isolation*, namely the identification of a particular fault. To this end, they must exhibit some kind of sensitivity -either directional or structural- to the properties of the fault. Moreover, residual filters must be robust to noise originating either from modelling errors or inherent sensor uncertainty. As one of the earliest model-based approaches, fault-detection literature offers extensive survey papers on parity relations [37, 38, 39]. Specifically, [37] reports on the process of residual generation for additive and multiplicative faults; moreover, implementation issues are also addressed.

The construction of the model is hard in most cases due to the large number of state variables involved. Another way of constructing a residual-based fault detection system is by using *bond graphs*. Bouamama et al. [40] use bond graphs [41] to eliminate the unknown system variables and provide an input-output mapping, namely the analytical redundancy relations. Bond graphs are compact representations of dynamic systems that can capture interactions between various subsystems in a common framework. This feature is pretty useful in practice, since most systems consist of various subsystems with different -yet coupled- dynamics. A limitation of the Bond Graphs is that they fail to model hybrid systems, i.e., systems with both continuous and discrete states. The Hybrid-Bond Graphs [42] have been employed to remedy this limitation [43].

Following fault-detection, isolation of the fault becomes relevant. As mentioned above, following residual generation, by any available model, the residuals are further processed by a linear filter in a way that promotes fault isolation(enhanced residuals). Appropriate choice of the transfer function of the filter eliminates the influence of noise [35]. Additionally, it imposes certain behaviour of the residuals in response to specific faults. An in depth description of the computation methods of such residuals -both for additive and multiplicative faults- can be found in [44]. The most common types of enhanced residuals are the following:

- Diagonal residuals : each dimension of the residual vector is sensitive to one fault only. In this way multiple faults can be isolated simultaneously
- Directional residuals : in this approach the fault response lies within a specific direction (e.g., line) in the residual space. Multiple faults can be isolated simultaneously if the respective directions are independent
- Structured residuals : the residuals are sensitive to a subset of faults. The dependency between fault-residual is encoded in a matrix that has as rows the residual elements and as columns the faults under consideration. An entry of value "1" in the ij-th position means that the residual i is sensitive to fault j. This induced fault matrix can be read in two ways: a) a row indicates the faults that can influence a specific residual element b) a column is the fault

signature of a specific fault; the set of residuals that are influenced. A fault is detectable if there is at least one non-zero entry in the corresponding column. Furthermore, it can be isolated, if its fault signature is unique within the fault matrix.

Observers: Luenberger observers are a well established way of reconstructing the internal state of a system from known inputs and outputs. The state space formulation can easily integrate additive faults as disturbances in the state equations. Multiplicative faults are often transformed to additive faults with time-varying disturbance-to state matrices [35]. The difference between the estimated and the actual state forms the error vector, which is later multiplied by a weight matrix to give the residuals. The gain of the observer as well as the weight matrix are chosen appropriately to achieve either of the following objectives:

- robustness to input disturbances
- sensitivity to predefined faults

There is a trade-off in the choice of the aforementioned matrices. Sensitivity to faults improves fault isolation but lacks robustness to disturbances. As an effort to remedy the above limitation, *unknown input observers* have been developed. This type of observer is explicitly designed to be robust to input uncertainty [45, 46]. Using full state unknown-input observers provides the designer enough flexibility for fault isolation. Some researchers use Kalman filters to estimate the state before the residual generator [47].

The gain of a fault-detection observer must provide stability and ensure that the error corresponds to a specific fault and not to noise. This is achieved by ensuring that error due to faults remains in direction in the state-space (a subspace). If the set of subspaces, corresponding to individual faults, are linearly independent, multiple faults can be detected in parallel; achieving as such *fault isolability*. Park et al [48] describe the procedure of designing a fault-detection filter based on full-state observers. In [49] a fault-detection filter is used to enhance aircraft reliability, by monitoring the sensors and the actuators of an aircraft. In regions of marginal

stability, *i.e.*, when the observer's eigenvectors are ill-conditioned, the filter becomes sensitive to parameter uncertainty. Filter robustness has been discussed in [50, 51]. [52] employs a game-theoretic approach to design a robust fault-detection filter.

Filter robustness has been approached from a completely alternative point of view; optimisation schemes have been applied to design fault-detection filters with minimum sensitivity to uncertainty, albeit with maximum fault-isolation sensitivity. Stoustrup and Niemann [53] provide a survey of optimisation-based approaches. Specifically, fault detection and isolation is formulated as a general robust control problem. It is the first complete discussion of both closed-loop control performance and fault-detection capability. [53] showed that there is a trade-off between control performance of the closed-loop system and performance of the reliability framework for systems with unknown parameters. [54] investigated fault detection for linear systems with modelling uncertainties using the parameter-dependent bounding approach and multiplier theory [55]. Authors in [55] report that such approach is less conservative than systems derived using the small gain theorem and general Lyapunov stability concepts. Minimisation to parameter disturbance is then posed as an optimisation problem subject to a Ricatti equation constraint. The yielded fault-detection algorithm is applied on a flight control system.[56] derived a complete fault detection and isolation algorithm using optimal estimation, where the objective is to build an optimal estimator that is stable in the presence of noise. Both an $\mathcal{H}_1$ and an $\mathcal{H}_2$ norm of the signal have been exploited.

Another observer based-approach, that deals with multiplicative faults in their original form, is the set-valued observer method [57]. The unknown parameters are assumed to lie in a convex polytope. The estimation of the state produces a set of possible states rather than a single value [58, 59]. This set is rarely convex, even if the set of previous states is. To remedy that, the convex hull of the resulting estimation set is computed. A fault is detected when the set of estimated states is empty; there is no possible explanation for the state evolution. Furthermore, a bank of Kalman filters is used for fault isolation. For every parameter in the convex parameter polytope, a Kalman filter is designed. Following the detection of a fault,

all the filters are activated.

There are two common observer-based architectures for fault isolation [60]

- Dedicated Observer Scheme: the s-th residual is sensitive to one specific fault and decoupled from all the others (strongly resembles the diagonal residuals described above)
- Generalized Observe Scheme: the s-th is residual is sensitive to all but the s-th fault

In the latter case the s-th fault is detected if all the residuals but the one associated with s are above a certain threshold.

A special case of optimisation methods is that of Kalman Filter; it merely uses a linear quadratic objective function. [61] introduced the Kalman filter as general method for generating residuals, namely *innovations*. In the presence of a fault, residual distribution exhibits a distinct bias; contrary to the healthy cases, where residuals are merely corrupted by white noise. In this way, fault detection is performed by testing the residual whiteness. Further statistical testing occurs on the mean and covariance of the residuals. Exemplary statistical tools pertain fault detection by residual evaluation are the Maximum-Likelihood (ML) method and the generalized likelihood ratio (GLR).

A well-known Kalman filter for fault detection is based on the multiple-model adaptive estimation principle (MMAE) [62]. In this approach the system dynamics are modelled by a stochastic linear model with uncertain parameters. The parameters, which include the faults, take values from a discrete, countable set. For each parameter permutation, a separate Kalman filter is instantiated; yielding a bank of filters. MMAE can be found in many aerospace application comprising navigation systems [63] and aircraft control systems [64]. Residual classification of the latter has been considered by [65]. The yielded scheme was applied to sensor and actuator failure in a flight control system. Since the number of imminent faults is known from before and is finite, respective mitigation strategies may be developed. Nevertheless, complete enumeration of faulty modes is unrealistic in many autonomous systems.

An increased number of possible faults would increase the computation cost to non-applicable levels. Moreover, in many systems the known-faults assumptions is not realistic.

Residual Evaluation After residual generation, the diagnostic system needs to reason about the importance of a potential deviation from zero. A simple way to do so is to use a threshold value, above which the difference from zero is considered significant. The choice of the threshold is not straightforward. If the threshold is set too low, the number of false alarms is likely to increase. To the contrary, a high threshold will fail to detect faults. A static threshold may be inadequate, if the residual generation is not inherently robust to input noise. Parameter shifts due to aging cause additional problems in the application of static thresholds. A way to mitigate this is to vary the threshold over time(adaptive thresholding). An early version of adaptive thresholds based on fuzzy logic [66, 67] is presented in [68]. Zhang et al. in [69] describe a method to adjust the thresholds on-line given an upper bound for the parameter uncertainty.

2.1.2 Statistical approaches

Model-based approaches have been used extensively for fault detection and isolation. There are many situations, however, where acquiring the dynamic model of the system is not straightforward. Several methods from the statistics literature have been employed for fault detection in the absence of a mathematical model [3]. Historical data are processed to extract feature properties for the nominal case (normal plant operation), as well as for the faulty case.

The fault detection problem is posed as a statistical test; where, the *null* hypothesis H_0 represents the faulty mode of operation, and the *alternative* hypothesis H_1 which stands for operation in absence of faults. The data used for this particular inference are similar to data used for maintenance; intrusive machine inspection is not required. A hypothesis test is chosen, providing as such an answer to the fault-detection problem. Exemplary papers on this approach are [70]. A more recent framework for fault diagnosis, namely the *structural hypothesis tests*, extends

single statistical tests to the case of multiple distinct faults. Another method, originating from quality control theory, has been widely applied: *Statistical Process Control* (SPC) measures the deviation between a reference (baseline) representing the normal operation and the actual measured state of the monitored agent. SPC provides the reference values, above which the system is considered as broken. [71] presents an application of statistical process control for damage detection.

Other examples of this paradigm are the Sequential Probability Ratio Test [72] and the Cumulative Sum Ratio Test [73]. The latter also takes into account incremental shifts that occurred in previous time instants. Hence,it is more effective in detecting slow evolving incipient faults. Another common method uses control charts [74], as in Statistical Process Control, to check whether the state variables are within the plant's in-control limits. Modern industrial processes have numerous variables that need to be monitored. Instead of constructing individual control charts for each, multivariate statistical metrics are used such as Hotelling's T^2 or the Q statistic [75].

One large limitation of all the above is the independence assumption for the process variables. This assumption is often too crude in real world applications. For this reason, tools from the field of multi-variate statistics have been used to eliminate this drawback. *Principal Component Analysis*(PCA) [76] maps the data to another space -often of lower dimension- using a linear transformation. The bases vectors of the resulting space are sorted with respect to their contribution in the data variance(first coordinate indicates the direction of most variance in the data). Since the aforementioned vectors are orthogonal,the data is uncorrelated in the transformed space. Dynamic PCA [77] has been developed to capture the series correlation along the time axis as well. Dynamic PCA implicitly makes a stationarity assumption, namely that process statistics do not change over time. Recently, Mina et al. [78] presented an extension that also accounts for drifts of the process variables' means. Another method for dimensionality reduction is the Partial Least Square(or Projection to Latent Structure) [79]. It is applied when some of the process variables(predictors) are used to predict the values of a set of

Quality variables. A transformation is computed such that the principle directions of the transformed space maximize the covariance between the predictors and the quality variables. Fault detection methods based on PLS can be found in [80].

After the detection, the data undergoes additional processing for the analysis of what actually caused the fault. One approach is based on the decomposition of the T^2 statistic to a number of orthogonal factors which are also statistical distances [81]. The factor responsible for the deviation of the residual from zero is identified. Next, the cause of the fault is isolated based on the way in which the relevant system variables interact. The problem with this method is the combinatorial explosion of the number of possible decompositions as a function of the cardinality of the state space. A more informed way of performing such a decomposition leverages causal relations between the process variables. To this end, Bayesian networks are used to encode the causality information that guides the decomposition. [81] present a Bayesian framework both for the detection and the isolation of the fault.

Clustering is a powerful and well studied way of grouping objects based on feature similarity. The algorithms, used for clustering, maximize feature similarity within the groups, as well as inter-group diversity. The outcome of a clustering algorithm is a set of diverse groups of objects, where each of those groups contains object that are similar with respect to the classification task. In the case of fault detection, the objects are the data samples and the clusters are associated with faults. Based on predefined features on the signals, clustering algorithms attempt to approximate a map between those features and the faults. Some early application of cluster analysis on fault diagnosis are discussed in [82]. One of the most common algorithms in the clustering literature is K-Means. A notion of "distance" is used to model the similarity between different clusters. Various distances can be found in the statistics literature, *e.g.*, Euclidean distance, Mahalanobis distance. More sophisticated metrics that generalize the notion of distance for distributions have been used for diagnostics (e.g., Kullback-Leibler divergence). In particular, distance functions for fault detection can be found in [82]. Additionally, Support-Vector-Machines [83] and their kernel based extensions [84] can be used to optimise the boundaries between

the clusters. In these approaches detection and identification happen simultaneously. Apart from standard distance metrics from statistics and machine learning, new metrics have also been developed, tailored to the traits of the diagnostic tasks. Specifically, [85] is the first to use a new distance metric, namely *quotient distance* for engine diagnostics. Besides distance metrics, the correlation of feature vectors has been employed as another similarity measure [86].

Some methods [87, 88] use Hidden Markov Models (HMMs) as a tool for fault detection. In its simplest form, a binary variable -indicating faulty or normal system behavior- is used as the HMM's state. The observation sequence of the HMM consists of appropriate features of the monitored signals. As new observations become available, the Viterbi algorithm [89] computes the most probable state sequence, given the observations. Moreover, the state variable can be augmented to a vector, whose coordinates are indicator variables of different faulty modes. In this way fault isolation can be achieved in parallel to fault detection.

2.1.3 Artificial Intelligence approaches

Artificial Intelligence methods provide rigorous tools for regression and classification that can be applied to the problem of machine diagnosis. In many occasions, artificial intelligence methods outperform standard approaches. The main shortcoming of these approaches is the need for *labelled* data: that is, data that have been recorded during both faulty and normal operation. These data are essential to train the algorithm. Part of the training can be done using artificial data, resembling normal operation as well as a set of faults of interest. Most researchers, however, use experimental data to train AI fault-detection algorithms. The most common algorithms in the literature are *artificial neural networks* (ANN) and *expert systems* (ES). Other AI algorithms include fuzzy logic systems, fuzzyneural networks (FNNs), neuralfuzzy systems and evolutionary algorithms (EAs).

Artificial Intelligence techniques have been employed as an alternative to statistical approaches. Numerous variations of Artificial Neural Networks were applied to the problem of fault detection. Neural networks consist of basic processing units

(neurons) which combined together form a complex structure, that can approximate non-linear functions. The need of training data to compute the network parameters is the main drawback of this approach when applied in practical fault detection problems [90], [91]. Following a supervised learning paradigm the user needs to explicitly feed the network with the target values (faults in this case). Self-Organized Maps have been used [15] as a remedy to this limitation.

ANN can also be used in a model-based, yet data driven framework. Adaptive Autoregressive Neural Networks are used to approximate the evolution of the system variables based on the data history. Furthermore, the networks can adapt to slow changes in the parameter space, providing robustness to parameter change due to machine ageing. In [69] the authors use such a network in parallel with a state space model to approximate the effect of multiplicative faults in the state estimation. The lack of physical interpretation of the net is often considered a significant drawback against the model-based approaches. Similarly to the model based methods described in section 2.1.1 ,a bank of ANNs [92] can be used to model the dynamics of the process under a finite set of faults. These often called fault estimators are activated after a fault detection event to figure out what went wrong.

Another prominent AI approach is based on expert systems [93]. Expert systems represent human knowledge, in a way that facilitates computer aided reasoning. One typical representative of such systems is based on *if-then* rules(rule-based reasoning) [94]. The problem with rule-based systems arises in the maintenance of the knowledge base. Incorporating a new rule while preserving the knowledge base's logical consistency is not trivial. This approach, however, has been proven quite effective in situations where expert knowledge is ubiquitous. It is often used in a complementary manner to other methods or when modeling or learning of the system process is prohibitive. An extension to rule-based systems is case-based reasoning [95]; relying on experience from previous situations to reason new observations.

The described approaches also suffer from limitations. Expert systems require extensive enumeration of possible machine states. For this reason, a vast number

of ES rules is required to cover the combinatorial number of possible states. Moreover, the computational cost increases with the number of rules. Neural Networks, in turn, are often hard to train and require large datasets. Moreover, the structure of the neural network is chosen by the network designer; this might influence the performance of the network significantly. Simple architectures might be unable to capture the dynamics of the phenomenon adequately; whereas, convoluted networks are often prone to overfitting. Moreover, neural networks provide no physical interpretation of the implicit computation represented by their respective neuron connectivity. [96] combined artificial neural networks with expert systems to infer the wear state of machine tools. Likewise, [97] used a combination of ANN and ES to create a diagnostic systems for gas turbines.

Some researchers investigated the use of Petri nets for machine diagnostics. Petri nets are bipartite graphs with two different types of nodes: states and transitions. Arcs (graph edges) connect states with transitions and vice versa; but, never two states or two transitions. It is a compact graphical representation between conditions and events [98]. [99] used a Petri net for fault detection of complex systems. [100] combined Petri nets with Kalman filter and [101] first used case-based reasoning with Petri nets on induction motors, outperforming conventional case-based reasoning fault detection systems.

2.2 Prognostics

While diagnostic systems are concerned with the detection and identification of a fault, *prognostics* address the problem of predicting a fault before it actually occurs. Based on these predictions, proper actions can be taken to avoid failure. Therefore, prognostics are of uttermost importance for an effective condition-based maintenance system. Compared to prognostics, diagnostics may look inferior or even redundant. Yet, in practice, the two systems run in a complementary manner. There are always cases where the fault can not be foreseen. Furthermore, diagnostic information can be used not only to develop better prognostic systems but also to improve the design of the monitored plant in the first place. Most of the

prognostic methods in the literature, focus on the computation of the *Remaining Useful Life* (RUL). RUL is the time interval for which a system operates without failure. An alternative approach is to assess the probability of fault-free operation up to a certain time horizon (e.g. next inspection interval)[3]. Following the consensus within the prognostics community, this section concentrates on methods for the computation of the remaining useful life. An example of the second approach can be found in [102]. Prognostic methods fall into the following categories:

- Model-based approaches
- Statistical approaches

In what follows, the current status of the prognostic research is reviewed.

2.2.1 Model-Based Approaches

Physics-of-Failure: Physics of Failure attempts to capture the breakdown mechanism of a component in a mathematical model [103]. This model is most often derived from first principles. In addition to the system dynamics the degradation phenomenon is also modelled. The derived model is used to determine the component's current health status and later to extrapolate in the future using trending analysis to predict the time of failure. Thresholding is used to discriminate between normal and faulty operation. Failure Effects, Mechanisms and Modes analysis (FEMMA) is employed to model the failure mechanism of a component [104]. To this end, the following steps take place: 1) identify the failure modes , 2) analyse the effects that failure modes have on the system. Failure modes are then sorted based on their impact on the system as well as on their frequency of occurrence. The top ranked failure modes are investigated further to acquire a detailed model of the failure dynamics. With such a model at hand, the current health status is computed and an informed prognostic estimate can be made. Accelerated aging techniques are often used for the first step of the FEMMA; to identify the way accrued damage influences the system. First,the variables that are indicative of the failure mechanism are extracted. Next, their relationship with the failure mechanism is assessed.

Based on the application, various empirical fatigue models (e.g. Coffin-Manson law), have been used to model failure dynamics. An example application on a MOSFET can be found in [105]. Some of the most common failure models used in electronics prognostics are reported in [106]. The resulting model is later used to trend the fault precursor variables to a given threshold that discriminates between normal and faulty operation.

Canaries: Another approach uses canaries for the prognosis of a failure [107]. In the past, miners used these birds due to their increased sensitivity in the presence of hazardous gases as a warning system in the mines. A canary is a similar device to the one to be monitored, mounted at a nearby location. In this way, the canary device is exposed at the same environmental conditions as the actual one. It is engineered to fail faster, thus providing an early indication of the degradation of the actual component. An acceleration factor, namely the ratio of canary failure time to system failure time, is used to compute the Remaining Useful Life of the component. Often several canaries with different accelerating factors are used to determine the time of failure of the system [108]. An application of this technique for the prognosis of Ball Grid Arrays can be found in [109].

2.2.2 Statistical Methods

In absence of a model for fault propagation, various statistical tools can be used to estimate the remaining useful life from data. Given the time evolution of specific fault precursor variables, regression can be employed to forecast their future evolution. Auto-Regressive Moving Average (ARMA) and Exponential Projection methods [110] are amongst the most common techniques. The accuracy of the prediction is limited, especially for distant prognostic horizons. Nevertheless, their simplicity of computation make them attractive in certain occasions. [111] uses an ARMA model for the prognosis of bearings in vehicles. To account for varying operating conditions as well as unit-to-unit variance random processes have been used for forecasting. Random Coefficient Regression, originally proposed by [112] a stochastic degradation model that consists of the actual, yet unobserved, degrada-

tion level together with a zero mean constant standard deviation additive term. In this frame of thought, further prognosis techniques use other stochastic processes (Wiener Process, Gamma Process) for forecasting. Gebraeel et al [113] developed a random coefficient regression method based on the Wiener Process (Standard Brownian Motion) where the coefficients are updated in real time in a Bayesian manner. Another class of prognostic methods is the covariate-based hazard. The model relates the failure of the component to certain monitored variables (covariates). The most common representative of this class is the Proportional Hazard Model [114]. In this model the hazard rate of the system is factorised into two terms. The first term is the baseline function and models the average life time of the component(often the Weibull distribution is used as a baseline). The second term accounts for the unit-to-unit manufacturing variance, varying operating conditions and any other parameter than can influence the degradation process. One advantage of the Proportional Hazard Model is that different covariates can be easily combined with a certain baseline function. In this way, one could analyse the influence of different covariates on the total hazard rate. Some example applications, specifically on prognostics can be found in [115]. One very common approach is to use Hidden Markov Models [116]. This model consists of a markovian sequence of hidden states. The states take values most often from a discrete set, representing the component's health status. The model also captures the probabilities of transitions between states. The state is hidden in the sense that it cannot be directly observed. The state can be inferred based on the observation sequence. Fault precursor variables are used for the observation sequence. The prognosis given an HMM model takes place as follows. Firstly, the current hidden state is estimated based on the observations hitherto. Given the estimation of the current state, the transition probabilities are used to estimate at which point in time the system will enter a faulty state with a certain probability. Approaches based on HMMs are reported in [3] [117]. One limitation of the HMMs is that the probability of staying in a certain state follows a geometrical distribution. This assumption is often too crude for practical applications. To remedy this, Hidden Semi-Markov Models [118] have been developed. In a Hidden Semi-Markov

Model the state duration is also treated as random variable; it is drawn from a probability distribution. The parameters of such distribution are included in the model parameter set and are learned during the training phase. HSMM have been proven to deliver better prognostic accuracy [119]. Another extension of Hidden Markov Models uses *Mixtures of Gaussians* for the representation of the emission probabilities of the observation sequence in the case of continuous variables. [120] presents a prognostic algorithm based on this concept.

Dynamic Bayesian Networks [121] can be used in the case where the state needs to include more than one variable. The same thing can be done with HMMs by assuming that the state is a tuple and the state space consists of all its possible values. This results, however, in a very complex transition probability matrix. Dynamic Bayesian Networks have the same expressive power, albeit they need fewer parameters to be defined [122]. For prognostics, the state can be augmented to take into consideration the operational and the environmental conditions [123]. In Muller et al. [124], the effect of the maintenance actions is also integrated in the computation of the remaining useful life. Another tool, which has been used in data-driven failure prognostics, is the Kalman filter. The filter is used to forecast the mean value of fault precursor variables based on a transition model of predefined order whose parameters are learned from training data. The order of the transition model plays a significant role in the accuracy of the results. The state vector of the filter can be augmented with the model parameters thus allowing for time-varying processes to be modelled [125]. For forecasting, the state transition model is applied recursively using the latest estimation of the state as initial conditions. In [126], the Kalman filter is used to track the time evolution of a crack in a tensioned steel band. Kalman filter makes a linearity (local linearity for the case of the Extended Kalman Filter) assumption for the transition model, i.e. the dynamics of the process. A generalization of the Kalman filter, namely the Particle Filter, has been used to approximate a probability density function for the remaining useful life. The particle filter also relaxes the Gaussian assumption of the Kalman filter. In the case of particle filters the target distribution is approximated by a minimal set of particles [127].

In general, prognostic approaches based on stochastic filtering(e.g. Kalman filter variations and Particle filters) are advantageous because they compute a probability distribution for the remaining useful life, rather than just a point estimate. This information provides a better insight to the decision making subsystem on its risk assessment. This information comes at the expense of computational efficiency.

2.3 Applications in Underwater Navigation

Underwater missions require information about the AUV's position and orientation (i.e., pose). Navigation systems are assigned with this task; namely, to estimate the robot's pose with respect to a fixed frame of reference. The main difficulty in underwater navigation arises from the absence of GPS. Water molecules absorb the energy of radio waves very efficiently. In this way, the penetration ability of the radio waves decreases; thus, making GPS reception impossible after a certain depth. For this reason, navigation systems rely on alternative technologies.

Inertial Measurement Units (IMUs) measure the vehicle's linear and angular velocity. To this end, IMUs employ an ensemble of linear accelerometers combined with three gyroscopes. Doppler Velocity Logs (DVLs) use the time-of-flight of acoustic waves to compute the linear velocity of the vehicle. Often, DVLs measure currents, too. Magnetic compasses provide information about the vehicle's orientation. Pressure sensors measure the vehicle's depth. Inertial Navigation Systems (INS) integrate the output of several sensor modalities (e.g. accelerometers and gyroscopes) to yield a complete navigation unit; which accounts for the estimation of the full pose of the robot.

Kinsey *et al.* [128] provide a thorough overview of hardware development for navigation. Sensor failures plague navigation systems to a vast degree. One standard solution to sensor failures is *sensor redundancy*; i.e., to equip a system with multiple sensors that measure the same quantity. [129] describes a setup of multiple sensors and the appropriate algorithmic framework for dealing with various types of failures. In robotics, however, redundancy is not always possible; power and space limitations may prevent the placement of extra sensors on the robot.

Advances in navigation instrumentation have led to a large variety of navigation sensors. The latter provide information for different dimensions of the vehicle's state (i.e. orientation, depth, velocity). Consequently, algorithms that can combine such heterogeneous information are required [130]. Moreover, several precise models have been presented in the literature [131]. Measurements and model predictions are combined to yield an improved state estimation. The Kalman Filter (KF) is frequently used for such a task. Notably, the Extended Kalman Filter (EKF) constitutes the common framework for numerous navigation algorithms [132, 133, 134]. Recently, more sophisticated algorithms have been used for navigation [135]. [136] addresses the problem of multi-rate sensor fusion using a bank of Kalman Filters; each operating for different combinations of sensors. [137] use factor graphs for asynchronous sensor fusion. A detailed review of multi-sensor fusion techniques can be found in [138].

To compensate for the absence of GPS, acoustic navigation techniques have emerged. Acoustic navigation employs a number of beacons, anchored at known positions [139]. The vehicle interacts with the beacons using acoustic signals. The response time of the beacons is used to compute the vehicle's position. Specifically, Long-Baseline Localisation (LBL; [140, 141]) utilise several beacons that are anchored on the sea floor. Short-Baseline Localisation (SBL; [142]) employ multiple surface mounted transceivers. The deployment of multiple beacons offers sensor redundancy to the positioning system. Finally, Ultra Short-Baseline Localisation (USBL) uses a single transponder mounted on a surface vessel [143]. USBL offers the advantage of low system complexity. Acoustic Navigation suffers from the presence of outliers due to multi-path propagation errors. [144] and [145] augment SBL navigation with a Kalman Filter to enhance the robustness of the system. [146] provide a good overview of acoustic navigation systems.

Acoustic Navigation offers the most accurate solution to the navigation problem hitherto. In some situations, however, the vehicle is used for exploration. In such cases, the vehicle's workspace is not defined a priori. Therefore, encompassing the region of operation with acoustic beacons is not always feasible. Mounting the

beacons on autonomous surface vehicles [147, 148, 149] can alleviate this problem. Nevertheless, the system gets more complex and requires additional software for station keeping and localisation of the surface vehicles. To spare the extra complexity, often the surface vehicle is manned (i.e. not autonomous). However, certain environments do not facilitate the presence of surface vehicles (e.g. exploration under ice, underwater cavity exploration). Adopting the navigation framework presented in this thesis does not exclude the use of acoustic navigation; to the contrary, the presented algorithms can encompass acoustic navigation as another measurement stream for the vehicle's state.

The Kalman filter and its derivatives combine a mathematical model with sensory information to yield an estimation of the robot's pose. To this end, KF requires information about the accuracy of both the model (process) and the measurements; the accuracy of each is reflected on the respective covariance. Within the standard KF formulation, the above covariances need to be selected by the user. Several attempts have been made to estimate the model and measurement covariances online. [150] employ Neural Networks to tune the process covariance in an online fashion. Similarly, [151] adapts the process covariance in real time within an Unscented Kalman Filter (UKF) formulation. [1] provide a filter that automatically adjusts the measurement covariance. In this way, the filter excludes outliers from the final state estimation. ORKF [152] generalises Ting's work to cases where the measurement noise is correlated.

Statistical tests assess whether a data sample comes from a particular distribution under a given confidence interval. Such a test, often referred to as a *compatibility* test, could be performed on the distribution of the residual (innovation); namely, the difference between the model and the measurement. Whenever a residual fails the test, the respective measurement is rejected as an outlier. This approach implicitly assumes that the model is correct, and the residual is due to a false measurement. Conversely, the navigation algorithm presented as part of this thesis asserts both the model and the measurement accuracy. Hence, when the model is not accurate enough, the algorithm may consider a measurement that generates a high residual;

since on this occasion, the residual is due to errors in the model prediction. At this instance, a compatibility based approach would falsely reject the measurement. Moreover, *innovation gates* [153] employ thresholding for accepting/rejecting measurements; inferring as such the health state of the sensor. Since gating introduces additional parameters to tune, namely the accept/reject threshold, it has not been considered as an alternative.

Often, machine learning has been utilised to improve the model's accuracy within a Kalman Filter. Most commonly, Neural Networks have been used to compensate for un-modelled non-linearities in the system's dynamics. In specific, [154] report such an algorithm for GPS navigation. In a similar fashion, [69] use Neural Networks to model the non-linear dynamics of a jet engine. Additionally, Gaussian Processes (GPs) have been employed by [155] to compute a corrective term for the dynamic model of a blimp. Similarly, the authors have used LWPR to compute a corrective term [156]. In this work, LWPR was used to model the full dynamics of the system. Using a non-parametric algorithm simplifies model training; i.e. no identification of the core hydrodynamic model is required.

Apart from sensors, thrusters for underwater vehicles are also susceptible to failure. For this reason, thruster failure detection constitutes a broad field of research [157]; ranging from localised inspection mission to long-running scientific expeditions [158]. To identify a degraded thruster, *model-based* diagnostic approaches are commonly used. The model is used to compare the runtime and the expected behaviour of the monitored components. As outlined in [159], modelling the behaviour of marine thrusters is a difficult task; mainly due to unmodelled non-linear phenomena as well as because of gradual deviation from the ideal behaviour over the component's lifetime. Other methods [160, 161] model the actual component using closed formulations, based on device's external and internal coefficients. In this thesis, modelling has been undertaken by a data-driven approach.

An utter thruster failure may lead to locomotion degradation; i.e., the vehicle may lose a degree of freedom if no redundancy is available in the thruster configuration. As a remedy, [162] compute the optimal vehicle trajectory for compensating an

actuator's loss; i.e., treating the vehicle as a nonholonomic system. Following failure detection, failure mitigation adapts the required subsystems to preserve platform's functionality to the new operational conditions. A well-known approach [163, 164] to overcome the effect of a degraded actuator is to modify the thrust allocation policy. The presented algorithm is based on [165]. [165] has been extended to incorporate a hidden state; i.e., the unobserved state of the vehicle. This extension can be viewed as a generalisation of the *switch* Kalman filter [166]. The difference in the Bayesian formulation lies on the rigorous representation, which eliminates the need of assumptions for computing the final state; to avoid combinatoric explosion during state computation.

On the modelling side, there is a distinctive shift in underwater navigation. Ever more researchers employ machine learning techniques. Indicatively, [154] used a neural network to compensate for unmodelled non-linearities in GPS navigation. [69] used neural networks to compute a corrective term for the dynamic model of an aircraft engine. [155] used a Gaussian Process to approximate a non-linear corrective term for the dynamics of a blimp. Similarly, [156] used Locally Weighted Projection Regression to compute an accurate model for underwater navigation.

Chapter 3

Dynamic Modelling

3.1 Introduction

Autonomous agents are expected to interact with a highly dynamic world. Interaction pertains *perceiving*, as well as *acting*; i.e., to deliberately modify the environment in which the agent operates. To decide how to act, the agent needs to reason about the world's state. The latter is often hard to be observed directly; hence, sophisticated *inference* mechanisms are employed to estimate the state as a function of measureable quantities. Based on the inferred state, the most beneficial action towards the completion of the agent's mission is selected. Inference about such an uncertain system, as is the world around us, may be daunting; albeit necessary for correct decision making.

To reason about the world, an autonomous system requires a suitable world representation (model). To detect hardware degradation, specifically, a model of the interaction between the autonomous agent and the environment is necessary. Deviation of the observed state, as measured by the sensors, from the model predictions is indicative of possible hardware failure. In what follows, a general approach towards modelling the dynamics of an autonomous system is presented. Finally, the method is applied to model the dynamics of an Autonomous Underwater Vehicle (AUV).

3.2 State-space Formulation

Every system can be described by a minimal set of time-varying parameters. This set of parameters, namely the *state* of the system, is sufficient to infer any other property of interest. Quite often in the literature, the state parameters are combined in a state vector: $\mathbf{x} \in \mathbb{R}^n$ s.t. $\mathbf{x} = [x_1, x_2, ..., x_n]^T$. The dimension n of the state vector $\mathbf{x}$ is equal to the order of the differential equation of the modelled physical phenomenon.

The differential equation mentioned above constitutes the *dynamic model* of the system; i.e., a mathematical representation of the physics of the system. The most commonly used representation for such models is the *state-space* formulation. Generally the state-space formulation comprises the state, as defined above, together with a set of input and output variables. The output variables, namely $\mathbf{y} \in \mathbb{R}^m$, are also known as *observations*; observations are often a linear combination of the state variables. On the other hand, the input variables $\mathbf{u} \in \mathbb{R}^d$ are used to *drive* the system (i.e., to take the system from one state to a desired, final state). All the vector forms of the variables described above are related by a first-order differential equation:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t) \tag{3.1}$$

$$\mathbf{y} = h(\mathbf{x}, \mathbf{u}, t) \tag{3.2}$$

where $t \in \mathbb{R}^+$ denotes the time, $f : \mathbb{R}^n \times \mathbb{R}^d \times \mathbb{R}^+ \mapsto \mathbb{R}^n$ is a vectored valued differential equation that controls the evolution of the state over time (dynamic model), and $g : \mathbb{R}^n \times \mathbb{R}^d \times \mathbb{R}^+ \mapsto \mathbb{R}^m$ is the measurement (observation) model.

Depending on the form of the functions f and g, two special cases of the above model arise : *time-invariant systems*; functions f and g do not depend on time t, and *linear systems*; functions f and g are linear with respect to the state $\mathbf{x}$ and the input $\mathbf{u}$. Many systems can be modelled by time-invariant dynamics; i.e., systems that comply with the definition of an *autonomous system*. Most natural phenomena,

however, are governed by non-linear dynamics. Nevertheless, due to the simplicity of the theory underpinning linear systems, oftentimes non linear systems are modelled using linear approximations. Specifically, the dynamics of a non linear system are linearised locally using a first-order Taylor approximation of functions f and g (see Equation 3.3). This yields a model that is linear in a region of the state-input space; around a specific point $(\mathbf{x}_0, \mathbf{u}_0)$:

$$F(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}, \mathbf{u})|_{(\mathbf{x}_0, \mathbf{u}_0)} + \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{(\mathbf{x}_0, \mathbf{u}_0)} (\mathbf{x} - \mathbf{x}_0) + \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{(\mathbf{x}_0, \mathbf{u}_0)} (\mathbf{u} - \mathbf{u}_0) \tag{3.3}$$

where $\mathbf{x}_0 = \mathbf{x}(t = t_0)$ and $\mathbf{u}_0 = \mathbf{u}(t = t_0)$. By repeatedly linearising the dynamics along the state-input trajectory $(\mathbf{x}(t), \mathbf{u}(t))$, one gets a *linear time-variant* system of equations that approximate the original non linear system along the designated trajectory $(\mathbf{x}(t), \mathbf{u}(t))$. The quantity $\left.\frac{\partial f}{\partial \mathbf{x}}\right|_{(\mathbf{x}_0, \mathbf{u}_0)}$ denotes the *Jacobian* with respect to $\mathbf{x}$ evaluated at $(\mathbf{x}_0, \mathbf{u}_0)$.

In this thesis, the full non-linear dynamics are utilised. The measurement function, however, is linear. Moreover, by definition the dynamics of an *autonomous system* to not depend explicitly on time; hence, functions f and g are independent of t. Discretisation of the dynamics equation using a first-order approximation of the state derivative (see Equation 3.6) in the left-hand side yields:

$$\mathbf{x_{k+1}} = f'(\mathbf{x}_k, \mathbf{u}_k) \tag{3.4}$$

$$\mathbf{y_{k+1}} = \mathbf{C}\mathbf{x}_k \tag{3.5}$$

where $f'(\mathbf{x}_k, \mathbf{u}_k) = f(\mathbf{x}_k, \mathbf{u}_k) \cdot \Delta t + \mathbf{x}_k$, $\mathbf{C} \in \mathbb{R}^{m \times n}$ is a matrix realisation of the linear measurement model, and $k \in \mathbb{N}$ denotes the discrete time instants; each of constant length Δt (seconds). In this representation (3.4), a first-order Markov assumption has been made implicitly; namely, that the state at time $k+1$ solely depends on the input and the state at time k. The first-order Markov assumption stems from the numerical approximation -also of first order- for the time derivative $\dot{x}$:

$$\dot{x} = \frac{dx}{dt} = \frac{x(t + \Delta t) - x(t)}{\Delta t} = \frac{x_{k+1} - x_k}{\Delta t} \tag{3.6}$$

The above numerical approximation for the derivative yields an error of $O(\Delta t)$[1]. Using higher-order approximation schemes for the time derivative $\dot{x}$ can lead to errors lower in magnitude (e.g., $O(\Delta t^2)$, $O(\Delta t^4)$). The higher-order schemes, however, break the first-order Markov assumption; i.e., the prediction of the model at time $k + 1$ will depend on all $x_{k-k'}$, where $(k' + 1)$, with $k' \in \mathbb{N}$, is the number of past states[2] required by the model; the number of past states depends on the order of the derivative approximation.

3.3 Model Specifications

In the previous section, the semantics of the state-space formulation have been defined. The assumptions that accompany non-linear autonomous systems, with discretized dynamics and linear measurement (observation) models have been thoroughly explained; leading to the general form of a dynamic system, as described in Equations (3.4), (3.5). The focal point of this section is to provide further details about the requirements of the dynamic model; subsequently, also of function f (see Equation (3.4). Moreover, the problem of modelling the dynamics will be cast to a non-linear regression problem, on which powerful tools from the machine learning literature can be employed; yielding an easy-to-tune, yet accurate mathematical model of the dynamics.

An approach to defining f is by using the physical laws that govern the phenomenon under consideration. By using physics, a dynamic model can be constructed, as a set of equations that relate the state of the system (possibly the derivatives of the state) with the input variables. This set of equations oftentimes comprises several parameters (e.g., mass, moment of inertia) that are required for a complete model to be defined. Therefore, such models are known as *paramet-*

[1] $\Delta t << 1$ in most systems of interest

[2] In this thesis it is assumed that the natural numbers $\mathbb{N}$ also include zero

ric dynamic models. The parameters of such models are estimated by performing identification experiments.

During identification experiments the system, which we want to model, is driven along random trajectories of the state-input space. The identification trajectories must be appropriately designed to excite all subtle non-linear phenomena that may be present; hence, revealing the complexity of the system's dynamics. While the system is driven along the identification trajectory, the values of the state and the input are recorded. The recorded state-input pairs will be subsequently utilised to elicit the parameters of the model.

As mentioned earlier, the physical laws specify the structure of the model; constraining as such the model to be a member of a specific class of functions (e.g. polynomials of n degree). By identifying the parameters that better match the data from the identification experiments, the complete model is specified; namely, a specific member of the aforementioned class of candidate models is chosen. In many cases, the physical laws governing the system are highly complex. As a result, the derivation of the dynamic model with this method may be cumbersome. For this reason, the dynamic model is often extracted by employing simplified versions of the physical principles of the phenomenon. In this way, the derivation becomes easier algebraically; albeit the accuracy of the model is compromised. Moreover, parametric models are quite restrictive: no matter how carefully the parameters are chosen, the model will always belong to a predefined class of functions. Had the initial choice of structure been wrong, the model will never achieve high accuracy, independently of the identification effort.

Another important consideration is the adaptation capabilities of the dynamic model. Although, the dynamics of autonomous systems do not depend on time, adaptivity is a desired feature for a dynamic model. This is due to the variability in operational conditions. For example, an AUV may behave differently as a function of water salinity, temperature, changes in the hardware configuration. An adaptive model affords the fast recalibration of the existing model. Standard parameter estimation techniques would require the repetition of the full identification experi-

ments. Moreover, an adaptive dynamic model will be able to accommodate changes in dynamics due to hardware degradation; either due to ageing or an abrupt defect.

There exist autonomous agents with high dimensional state spaces. The state space dimensionality often imposes additional requirements for the performance of the dynamic model. The high state dimensionality may render the model slow for real-time usage. As part of a motion controller of a robot that is intended for manipulation, for example, the model may need to be queried for state estimations several times per second. Therefore, the dynamic model must compute the state estimation really effectively.

There is a trade-off between model complexity and model accuracy. Parametric models require decision on the structure of the model. This may aggravate the peak accuracy of the model. Moreover, adaptivity is a desirable trait for the model; as well as, decent scaling to high dimensional state spaces. Advances in machine learning present an elaborate alternative to parametric dynamic models, which respects all the outlined requirements.

Specifically, equation (3.4) can be considered as a non-linear regression problem with $\mathbf{x}_{reg} = (\mathbf{x}_k, \mathbf{u}_k)$ as the independent variable and $\mathbf{y}_{reg} = \mathbf{x}_{k+1}$ as the dependent variable (see Appendix A). There exist powerfull non-parametric algorithms for non-linear regression. By using such methods, the engineer does not need to specify in detail the structure of the model; conversely, the space of candidate models comprises all possible classes of functions. Moreover, some regression algorithms can be trained incrementally. Hence, the model is trained constantly as more data becomes available; endowing in this way the dynamic model with adaptivity. Furthermore, by exploiting dimensionality reduction concepts (e.g., Principal Components Analysis), machine learning algorithms can handle systems with high dimensional state spaces.

A non-parametric algorithm that complies with all the above specifications, namely Locally Weighted Projection Regression (LWPR), will be used throughout this thesis whenever a dynamic model is required. LWPR is easy to train, accurate, adaptive and can handle high dimensional state-input spaces with constant com-

putational complexity for each query. The algorithm is presented in full detail in Appendix A.

3.3.1 LWPR hyperparameters

The training process of LWPR is regulated by a set of hyperparameters. The value of the hyperparameters needs to be decided prior to training the algorithm. The purpose of this section is to provide insight into the role of the most important hyperparameters. Following that, a systematic optimisation procedure will be presented; eliminating, in this way, the need to tune the hyperparameters manually.

As described in Appendix A, the domain of a target function is partitioned in small neighbourhoods (receptive fields) where the function is approximated locally by a linear model. A receptive field is parametrised using a kernel that is centred at a specific point. The width of the kernel defines the activation region of the local model. When a new point requires a new local model (i.e., it does not activate any existing local model), a new neighbourhood is created, as explained in the preceding section. The initial width of the kernel is defined by the parameter $init\_D$. Specifically, the kernel's width is inversely proportional to $init\_D$. The width of the kernel will be adjusted online as more data become available. A stochastic gradient descent is used to adjust the width of the kernel (see Equation (A.7)). The step of the stochastic gradient descent, namely $init\_alpha$, can also be tuned by the user. Additionally, the online adaptation of a receptive field's width, performed by the stochastic gradient descent, can be switched off by setting $update\_D = 0$. In this case, all kernels have equal width; as indicated by the parameter $init\_D$.

The kernel's width is of uttermost importance to the performance of the method. Small values for $init\_D$ yield wide receptive fields. Depending on the level of non-linearity of the target function, wide receptive fields may not be appropriate; a larger value of $init\_D$ may be required. Conversely, large values of $init\_D$ result in very small receptive fields. In this way, more local models are needed to cover the target function's domain. For a sufficiently large value for $init\_D$, one local model may be fitted for each training point. Obviously, this situation must be avoided. For

this reason, a regularisation term is introduced. This regularisation term, namely *penalty*, penalised infinitely small receptive fields.

A few other parameters are also important to achieve the desired behaviour. Such parameters include w_gen, $init\_lambda$, $final\_lambda$ and tau_lambda. Firstly, w_gen is an activation threshold. When a new point becomes available, a new receptive field is created if the maximum weight of the activated models is below w_gen. The rest of the preceding parameters pertain to the adaptive behaviour of the algorithm. As mentioned earlier, a forgetting factor defines the influence of a new point in the internal parameters of the activated models. In the beginning, the forgetting factor is equal to $init\_lambda$. The forgetting factor changes gradually to $final\_lambda$ using simulated annealing with tau_lambda as the annealing constant. Lastly, $norm\_in$ scales the input to similar ranges, to improve the learning process of the algorithm. Table 3.1 summarizes the most important parameters of the LWPR algorithm:

Table 3.1: Tunable LWPR Parameters

Parameter	Explanation
$init\_D$	initial width of a RF
$init\_alpha$	step of the stochastic gradient decent
$update\_D$	on-line adaptation of width on/off
w_gen	RF generation threshold
$init\_lambda$	starting value for the forgetting factor
$final\_lambda$	final value for the forgetting factor
tau_lambda	annealing constant for the forgetting factor

3.3.2 LWPR Training

In this section, a disciplined way for choosing the training hyperparameters is suggested. The procedure is based on standard machine learning practices, used to train several algorithms in a *supervised* manner. After the training hyperparameters have been chosen, the stochastic gradient descent can be executed incrementally. In this fashion, LWPR will be able to continuously incorporate new training samples. Even for incremental algorithms, however, it is habitual to first train a model using a

large set of training data (batch training mode). After batch training, the resulting model can be refined by continuous adaptation.

To train LWPR, one needs to gather several data samples as input-output pairs; namely, $(\mathbf{x}_{train}, \mathbf{y}_{train})$. The whole of these data samples constitutes the initial available dataset. This dataset is then split into three smaller datasets:

- training dataset; holding the 80% of the original dataset. This data set will be utilized to train all the candidate models, as well as the final one (after the optimization of the model's hyperparameters)
- cross-validation dataset; this is used to optimise the LWPR hyperparameters)
- test dataset; this is used to evaluate the performance of the final model on unforeseen data.

Both the cross-validation and the test data sets hold 10% of the original data.

The training occurs as follows: for each of several combinations for the hyperparameters θ, a model is trained on the training dataset. The performance of each of the resulting models is evaluated on the cross-validation dataset. The hyperparameters of the model with the highest performance on the cross-validation dataset are chosen as *optimal*. The performance of the optimal model is then evaluated on the test dataset. The performance on the test dataset indicates the generalisation capability of the model; i.e., how well the model computes the target function for input data that have not been part of the training dataset (unforeseen data). The process of tuning the hyperparameters is often mentioned as *model selection*. Model selection can be easily cast as an optimisation problem:

$$\theta^* = \arg\min_{\theta \in \Theta} J(\mathcal{D}, \mathcal{M}) \tag{3.7}$$

where θ^* holds the optimal hyperparameters, Θ is the hyperparameter space, J is a performance metric that quantifies the prediction accuracy of model $\mathcal{M}$ on the data set $\mathcal{D} = (\mathbf{x}_{train}, \mathbf{y}_{train})$. The performance metric used in this case was the normalized Root Mean Square Error (nRMSE), as defined in Equation 3.9.

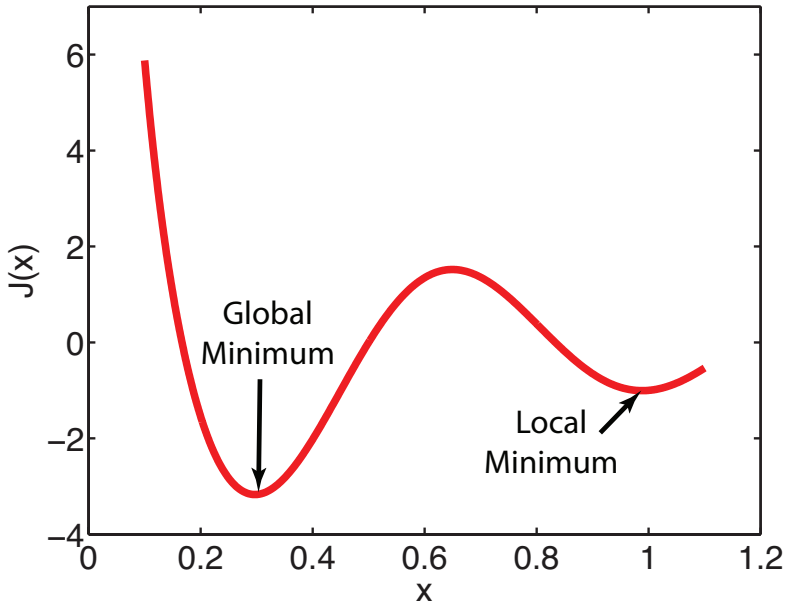


Figure 3.1: The problem of local minima in optimisation

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (x_1^{(i)} - x_2^{(i)})^2}{n}} \tag{3.8}$$

$$nRMSE = \frac{MSE}{x_{max} - x_{min}} \tag{3.9}$$

A general approach towards optimisation of a cost function $J(x)$ follows the gradient (or the negative gradient depending whether it is a minimisation or a maximisation problem) of the cost starting from an initial choice for the independent variable x. This paradigm is only valid in cases where the cost function is *convex*[3]; hence, it only has one global extremum (either minimum or maximum). This does not apply for non-convex functions; i.e., several local extrema can be found (see Figure 3.1). To this end, a more sophisticated optimisation approach is needed for non-convex cost functions.

The convexity properties of the cost function that is used for choosing the hyperparameters are not known precisely. Nevertheless, it is safe to assume that it is highly complex and most probably non-convex. Therefore, special attention is required to ensure that the potential solver does not get stuck in *local minima*. To

[3] *convex* for minimisation problems and *concave* for maximisation problems; for simplicity both are referred to as *convex optimization*.

avoid that, the cost is optimised by employing a multi-start search algorithm. A multi-start algorithm utilizes a local convex optimization algorithm to solve the same problem several times in parallel, with different starting points. The multi-start algorithm return the smallest of the local minima, as computed by each local convex solver, as the global minimum. Choosing a sufficient number of solvers reduces the the probability of convergence to local minima significantly.

Model selection for LWPR has been automated using the above formulation; i.e., treating the choice of hyperparameters as an optimization problem. The model selection software depends on Matlab's Global Optimisation Toolbox (part of which is the multi-start solver). Following the instructions in the LWPR tutorial [167], the parameter *init_D* is tuned first with the online adaptation of the kernel's width switched off. Next, the online adaptation is switched back on and the procedure is repeated to tune the rest of the parameters (*init_alpha*, *penalty*).

3.4 Experimental Evaluation

In this section, the previously outlined theory is applied to model the dynamics of an actual autonomous system. For this purpose, an Autonomous Underwater Vehicle (AUV) has been used. Specifically, *Nessie* (Figure 3.2) is a torpedo-shaped vehicle, developed within the Ocean Systems Lab. Nessie possesses three pairs of thrusters (see Figure 3.3). One pair of thrusters handles acceleration along the surge dimension. The lateral pair of thrusters controls the sway acceleration, and the last pair is responsible for adjusting the depth (heave) of the vehicle. The same pairs are capable of generating moments along the respective axes (i.e., roll, pitch, yaw), by applying forces in opposite directions[4]; thereby controlling the angular acceleration of the vehicle. This particular thruster configuration does not allow control of the roll of the vehicle; i.e., rotation along the longitudinal axis of the vehicle. Apart from the thrusters, Nessie holds several navigational sensors: a *Doppler Velocity Log* (DVL) uses the time-of-flight of a modulated array of acoustic signals to compute the linear velocity of the vehicle. A *Fibre Optic Gyrometer* (FOG) measures the angular

[4]moments are also exerted to the vehicle by unequal parallel forces within a pair

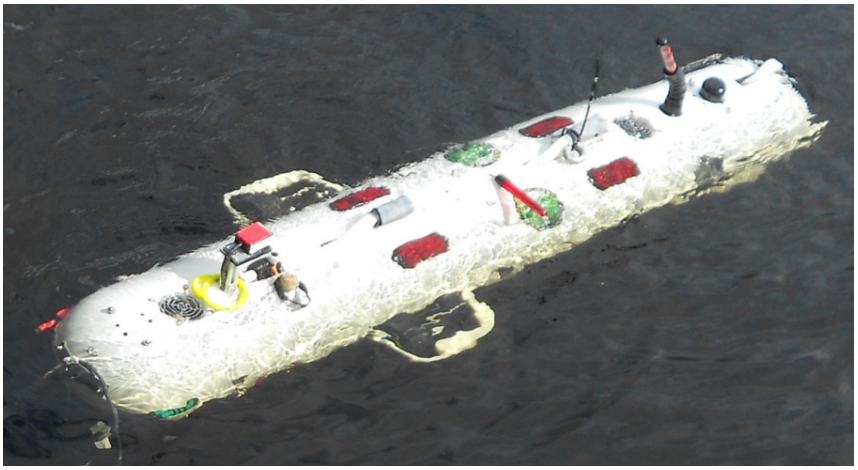


Figure 3.2: Nessie is the main research platform of the Oceans System Lab. It is a hover capable torpedo shaped AUV with a variety of sensors that are used for navigation (DVL,Gyro,Compass) as well as a Blueview forward looking sonar for perception.

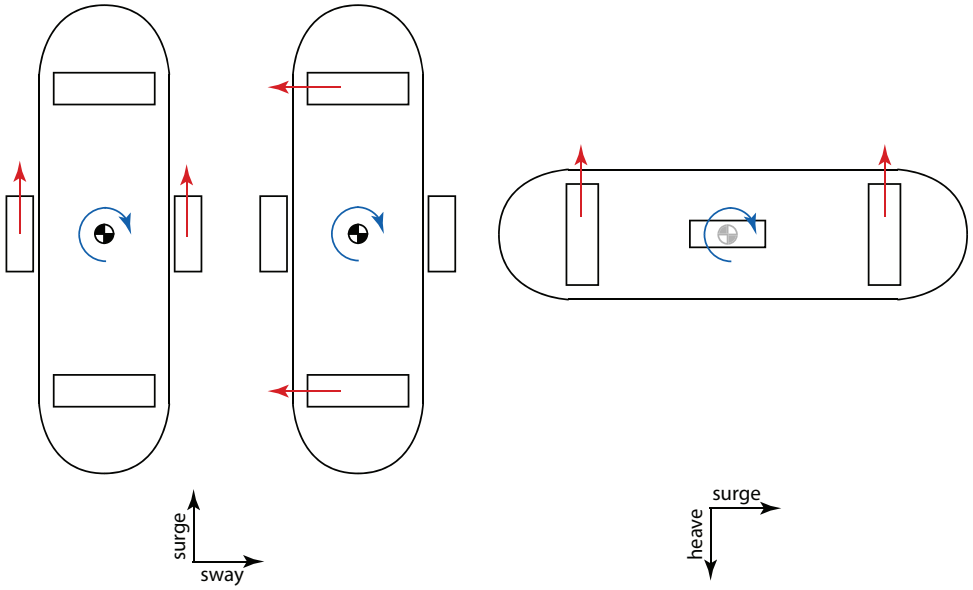


Figure 3.3: illustration of Nessie's thrusters

acceleration along the yaw dimension. A *magnetic compass* provides the orientation of the vehicle with respect to a fixed inertial frame, and finally a *pressure sensor* indicates the depth of the vehicle. Table 3.2 summarises the navigation sensors that are available on Nessie.

To gather data from the navigation sensors, Nessie was driven manually on a closed-loop trajectory. The experiments took place in a wave tank within Heriot-Watt University. Measurements from the FOG, the DVL and the pressure sensor, as well as the thrusters, were recorded using the *Robot Operating System* (ROS). Different trajectories were recorded for the training and the validation of the algo-

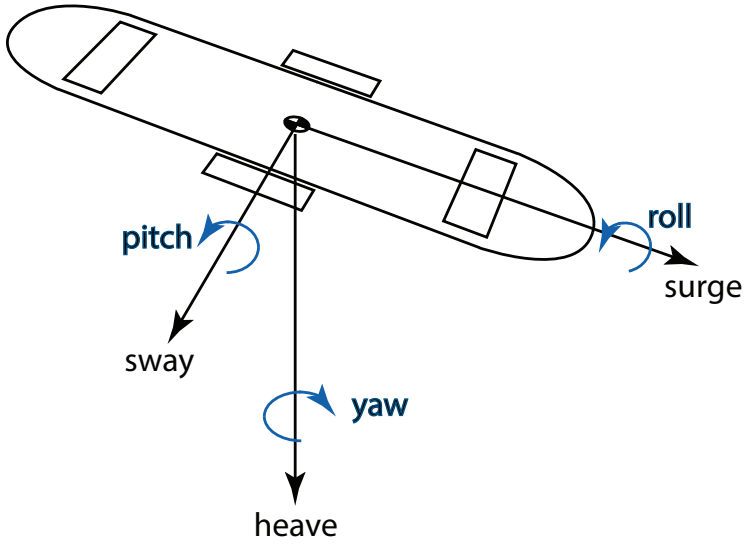


Figure 3.4: illustration of Nessie's dofs

Table 3.2: Nessie's navigation sensors

DVL	Teledyne Explorer PA
FOG	KVH DSP-3000
Compass	TCM 6
Depth Sensor	Keller Series 33X

rithm.

The input to LWPR is the current state $\mathbf{x}_k$ and the input to the six thrusters $\mathbf{u}_k$, whereas the output is the state of the vehicle at the next time instant ($\mathbf{x}_{k+1}$). Specifically, the state in this experiment consists of the *surge*, *sway*, *heave* velocities and the *yaw rate*. *Roll* and *pitch* were not controlled during the experiments. Accidental excitation of roll and pitch has been avoided by executing relatively smooth trajectories.

3.4.1 Order of the dynamic model

In the beginning of the chapter, the general form of a discrete dynamic model has been described (see Equation (3.4)). As already mentioned, this form implicitly assumed that system dynamics are first-order Markovian; i.e., the state at a future instant $k+1$ depends only on the state and the input at time k. In what follows, the validity of such an assumption is tested experimentally.

To this end, the performance of several dynamic models has been compared. The

input to each dynamic model comprises the commands to the vehicle's thrusters, as well as the n previous states of the vehicle. The number of past states may influence the performance of the model. The goal of this section is to test the hypothesis that the dynamics of Nessie are first-order Markov.

Equation 3.4 is a discrete realisation of a continuous differential equation of the form: $\dot{x} = f(x, u)$. As mentioned earlier, the order of the derivative approximation influences the number of past states n that appear in the discretised version of the dynamics. In order to choose a value for the parameter n, various models were evaluated on a cross-validation set. For each value of n, ten models were trained. The performance of the latter was averaged to yield an expectation for the normalised RMSE (Equation 3.9) for each value of n. The resulting learning curve is illustrated in Figure 3.5. The models with $n = 4$ perform best.

Further insight for the choice of n may be gained by exploiting the notion of *partial autocorrelation* from the Time Series Analysis literature. Figure 3.6 depicts the partial auto-correlation of the surge velocity in one of the training trajectories. Partial auto-correlation $\alpha(n)$ is the correlation between the samples X_k and X_{k-n} of a stationary time series X [168]. In this case, the assumption of stationarity is not valid. Nevertheless, the auto-correlation has been used as empirical evidence to underpin the choice of n. As shown in Figure 3.6, the influence of the past states on the current state is negligible for $n > 4$. This constraint is quite conservative, since it doesn't consider the information in the input time series.

In practice, the performance gain from choosing $n = 4$ rather than $n = 1$ is negligible compared to the increased computational cost. Moreover, the increased performance for slightly larger n may be due to some kind of filtering that occurs implicitly by using previous states. Consequently, for the sake of simplicity and performance $n = 1$ was the final choice for the dynamic model.

3.4.2 Model Perfomance

After having decided on the order of the model (see section 3.4.1), the LWPR model has been trained as described in section 3.3.2. Firstly, the hyperparameters have

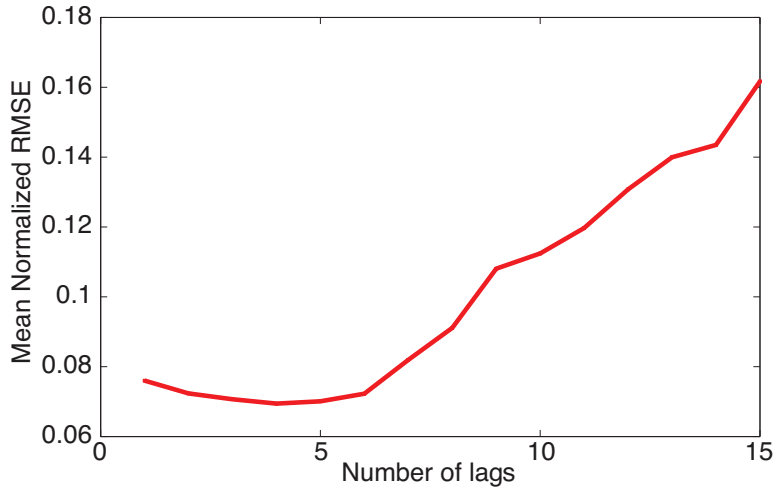


Figure 3.5: Normalized Mean Square Error (nRMSE) as a function of n; the number of previous states that are used as input to the model. For each value of n, ten models were trained on random permutations of the training set. Next, the nRMSE was computed for all the models using the cross-validation set. The geometric mean of each group of models was plotted against n. The resulting learning curve indicates the number $n = 4$ as the optimal choice.

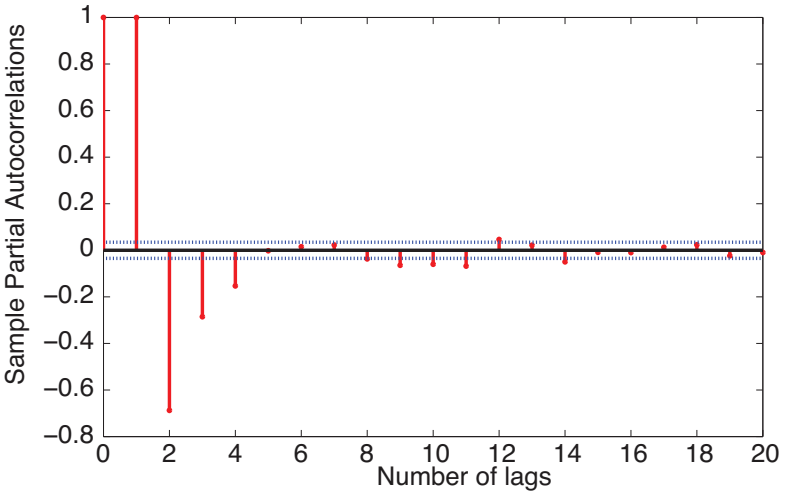


Figure 3.6: Partial auto-correlation for the surge dimension. In time series analysis, the partial auto-correlation $\alpha(n)$ is the correlation between the samples X_k and X_{k-n} of a time series X. The partial auto-correlation is used to define the order of auto-regression in an ARMA model; namely, the number of past samples used to forecast the value of a process. For $n > 4$ the influence of the n-th sample becomes negligible.

been chosen, using the cross-validation datasets. The resulting hyperparameters are summarised in Table 3.3:

Table 3.3: LWPR Hyperparameters after optimisation

DoF	init_D	init_alpha	penalty
surge	190	6	0.0001
sway	130	11	0.0001
yaw	32	22	0.0001

A reasonable estimation of the generalisation capability of the model is obtained by computing the performance of the resultant model on the test dataset. The test dataset consists of data samples that have been used neither for training nor for optimisation of the hyperparameters. Therefore, the test dataset is presumably indicative of the queries that the model will need to answer in practice.

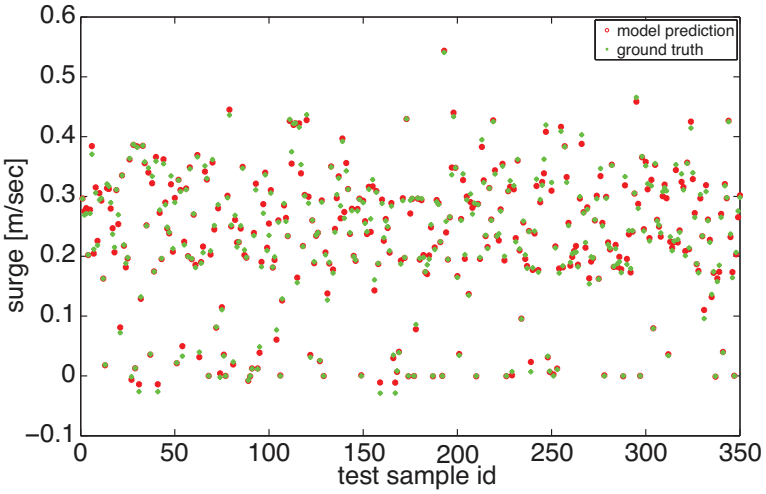


Figure 3.7: Comparison between the actual surge velocity and the predictions from the hydrodynamics and from our method.

Figures 3.7, 3.8, 3.9, 3.10 illustrate the predicted velocity along the surge, sway, heave and yaw respectively. The performance, in terms of the normalised Mean Square Error, can be found in Table 3.4

Table 3.4: Cross Validation Statistics

Dimension	*surge*	*sway*	*heave*	*yaw rate*
nRMSE (%)	1.21	2.01	1.32	1.30

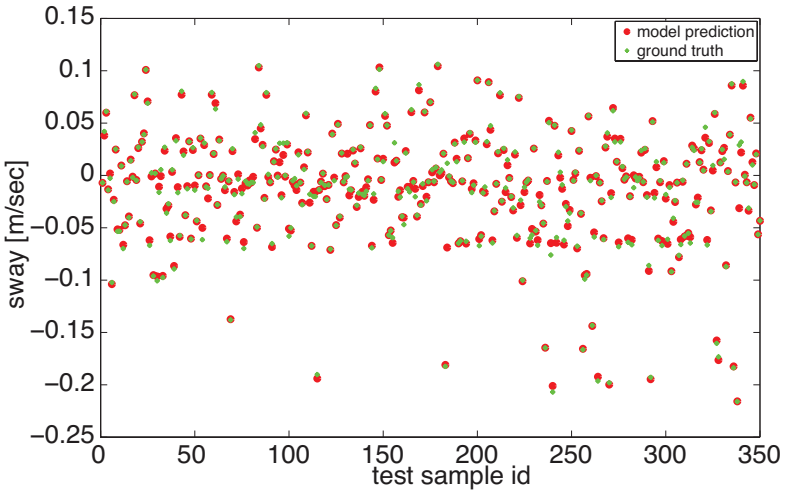


Figure 3.8: Comparison between the actual sway velocity and the predictions from the hydrodynamics and from our method.

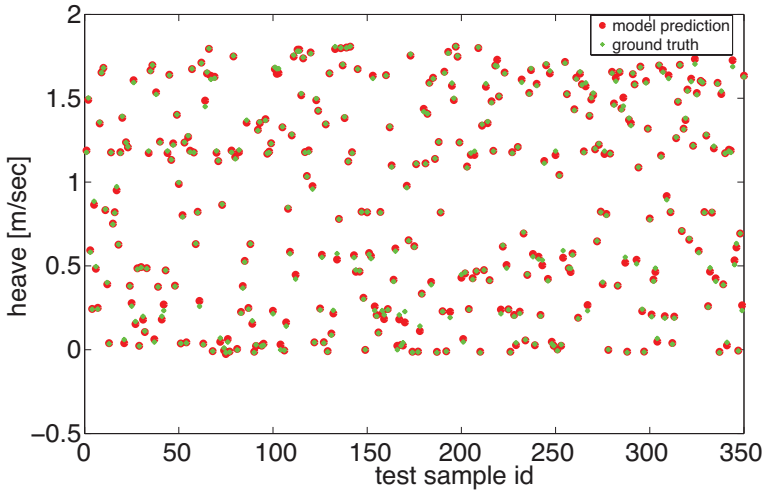


Figure 3.9: Comparison between the actual sway velocity and the predictions from the hydrodynamics and from our method.

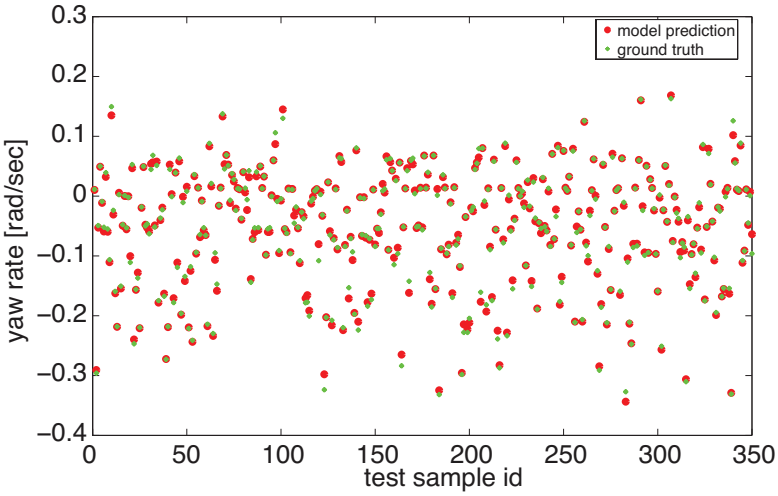


Figure 3.10: Comparison between the actual yaw acceleration and the predictions from the hydrodynamics and from our method.

Chapter 4

Robustness to Sensor Failures

All autonomous agents are equipped with various sensors that measure physical quantities of interest. The information from the sensors is used to yield an estimation of the state of the world; specifically, the part of the world that is relevant to a prescribed task (mission). Based on this estimation, the autonomous agent deliberates how to proceed towards mission completion. Next, the result of this deliberative process, namely the "plan", is executed.

In most applications the environment is dynamic; i.e., the operational conditions are subject to change. In this case, the agent needs to adapt the original plan to account for the new condition. The agent's reaction time to changes in the environment may vary significantly. Regardless of the reaction time, long-term autonomy entails sophisticated inference mechanisms to cope with the dynamic traits of a real-world environment. It is of profound importance to ensure that inference, as described above, is based on sound information.

A sensor failure constitutes an inadvertent condition at which the sensor provides wrong measurements; i.e., the measured value differs from the actual value of the quantity of interest, due to an unforeseen technical implication. If based on such sensors, inference will not reflect the correct state of the environment; hence, the computed course of action will either lead to failure or suboptimal performance. Common types of sensor failures include:

- the sensor is stuck at a constant value

- the sensor measurement is corrupted by random high-variance noise (outliers)
- the sensor goes offline; i.e., the sensor does not send any measurement updates

The goal of this chapter is to develop an algorithm that estimates the state of an autonomous agent in the presence of sensor failures. The integrity of each sensor is assessed at runtime. When the sensor integrity is compromised, i.e., in the case of a defective sensor, the algorithm detects the failure and disregards future information originating from that particular sensor. To this end, an accurate model of the agent's dynamics is utilised within a Bayesian Filter.

Generally, Bayesian Filters comprise two distinct steps: a *prediction* step and an *update* step. The prediction step uses a dynamic model (*transition* model) to predict the state. The update step leverages measurements from a set of sensors to correct the state estimation of the prediction step. Both model estimation and measurement update output probability distributions. Often, Gaussian distributions are used for that matter. Using Gaussian distributions is merely for convenience, and not a limitation of the Bayesian Filtering framework. Gaussian distributions are defined uniquely by the *mean* and the *covariance* (assuming a multivariate distribution - *variance* for the one-dimensional case). Consequently, a Bayesian Filter updates, in fact, the mean and the variance of the state probability distribution.

Variance (covariance in the multivariate case) quantifies how concentrated a random variable is around its mean. The more concentrated around the mean, the more reliable the estimation of that value. A Bayesian Filter combines model predictions with measurements, as mentioned earlier, by taking into account the respective covariances. Covariances are used to compute weighting coefficients for each state prediction, reflecting the corresponding accuracy. Finally, all predictions are combined in a weighted sum, to yield the final state estimation of the filter. In this way, the final estimation is biased towards the most reliable source of information.

Figure 4.1 depicts an example, at which information from two disparate "sensors" needs to be combined. The two distributions are parametrised as Gaussians with parameters (μ_1, σ_1^2) and (μ_2, σ_2^2) . The final estimation in this toy example is a weighted average of the means of the distributions, weighted as a function of the

respective *precision* (i.e., inverse covariance):

$$\mu_{final} = \frac{\sigma_1^2 \mu_2 + \sigma_2^2 \mu_1}{\sigma_2^2 + \sigma_1^2} \tag{4.1}$$

The higher the variance of a measurement the less its influence on the final result. Taking the limit of Equation 4.1 with $\sigma_1 \to \infty$ yields the following expression for μ_{final}:

$$\mu_{final} = \lim_{\sigma_1 \to \infty} \frac{\sigma_1^2 \mu_2 + \sigma_2^2 \mu_1}{\sigma_2^2 + \sigma_1^2} = \lim_{\sigma_1 \to \infty} \frac{\mu_2 + \frac{\sigma_2^2}{\sigma_1^2} \mu_1}{\frac{\sigma_2^2}{\sigma_1^2} + 1} = \mu_2 \tag{4.2}$$

Conversely, when $\sigma_1 \to 0$, $\mu_{final} = \mu_1$. Moreover, for $\sigma_1 = \sigma_2$, μ_{final} is simply the arithmetic mean (average) of the two values.

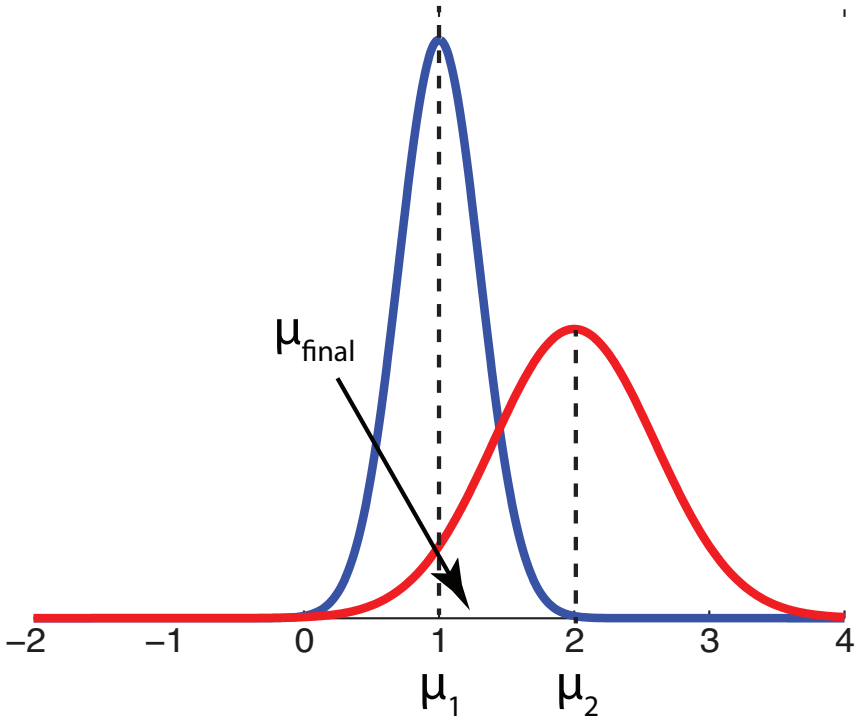


Figure 4.1: Example of mixtures of Gaussians

This simple mathematical analysis showed how the variances control the way that the information is combined. Therefore, the relative magnitude of the variances of the two filtering steps is crucial; it regulates which source is considered more trustworthy during the state computation. For robustness to sensor failures, an algorithm

adjusts the respective magnitudes of the variances, in a way that the model takes over in the case of a sensor failure. To this end, the algorithm treats the covariance of the measurement as a *hidden* variable; i.e., a variable that cannot be directly observed (measured). Instead, the measurement variance is sampled appropriately from a probability distribution. The model covariance is either kept constant or changes as well; estimated by applying the confidence interval computation as described in Section A.3.2. In this way, the relative importance of the measurement is not constant, but it changes dynamically based on the observed changes in the world. In what follows, a general method for devising such a Bayesian Filter is described. The showcase for testing the algorithm is underwater navigation; namely, the estimation of the position and orientation (pose) of an autonomous underwater vehicle with respect to a known frame of reference.

4.1 Bayesian Filtering

In Bayesian filtering, the objective is to estimate the *conditional* probability distribution of the hidden variables x (i.e., the state) given the observed variables y (i.e., measurements); namely, compute the conditional probability distribution $P(x|y)$. By definition, $P(x|y)$ can be written as:

$$P(x|y) = \frac{P(x,y)}{P(y)} = \frac{P(x,y)}{\int_{-\infty}^{+\infty} P(x,y)dx} \tag{4.3}$$

where $P(x|y)$ has been rewritten as a function of $P(x,y)$, i.e., the *joint* probability distribution of the variables x, y. Given $P(x,y)$, at least theoretically speaking, the conditional probability distribution $P(x|y)$ is computed. A powerful framework for modelling the joint probability distribution of a set of random variables is that of *Probabilistic Graphical Models*. Specifically, *Bayesian Networks* provide an elegant way of capturing conditional dependencies between random variables. A random variable is denoted as a circular node in the network. An arrow (edge) connecting two nodes, indicates a conditionally dependency between the two random variables.

Shaded nodes are random variables that are observed (measured), whereas standard circular nodes are considered to be hidden. With the above definitions, the semantics of Bayesian Networks have been provided.

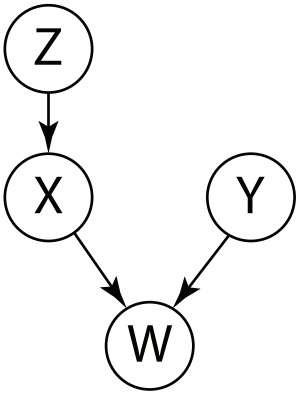


Figure 4.2: Sample Bayesian Network

For example, Figure 4.2 shows a simple Bayesian Network. The network is a graphical representation of the joint probability distribution $P(X, Y, Z, W)$. The Bayesian Network is utilised as a set of assumptions, which facilitate to break down the joint probability distribution to simpler expressions. To this end, the *chain rule* for Bayesian Networks is required: A random variable is conditionally independent of all its non-descendants nodes in the graph given the value of all its parents [169]. Mathematically, the chain rule is formulated as follows:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|\text{parents}(X_i)) \tag{4.4}$$

By applying the chain rule on the network in Figure 4.2, the joint probability distribution can be decomposed as a product of *factors* (factorisation):

$$P(X, Y, Z, W) = P(W|X, Y)P(X|Z)P(Z)P(Y) \tag{4.5}$$

After the individual factors, namely $P(W|X, Y)$, $P(X|Z)$, $P(Z)$, $P(Y)$, are defined the probability of any combination of the values of the random variables can be computed.

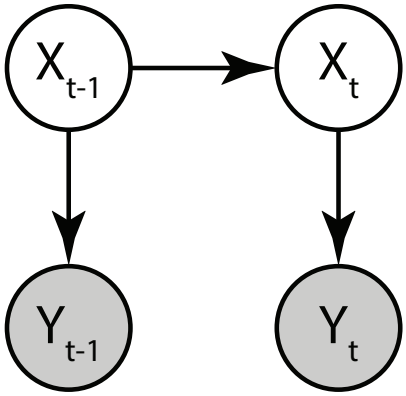


Figure 4.3: A *Hidden Markov Model* (HMM) depicted as 2-slice Dynamic Bayesian Network

A special case of Bayesian Networks is that of *Dynamic Bayesian Networks* (DBN). DBNs relate random variables at different instances in time. Figure 4.3 shows a common DBN, known as *Hidden Markov Model* (HMM). The random variable X_t is the hidden state, which evolves over time following some dynamics. The random variable Y_t is the observation at time t; i.e., a measurable quantity that provides implicit information about the state. The graph relates the random variables between time instances $t-1$ and t. Assuming that $\mathbf{X} = \{x_i\}_{i=1}^{k}$ and $\mathbf{Y} = \{y_i\}_{i=1}^{k}$, the joint probability of all states and observations across time is written as follows:

$$P(\mathbf{X}, \mathbf{Y}) = \prod_{i=1}^{k} \overbrace{P(y_i|x_i)}^{update} \prod_{i=1}^{k} \underbrace{P(x_i|x_{i-1})}_{predict} P(x_0) \tag{4.6}$$

The terms in Equation 4.6 directly correspond to the two distinctive steps of a Bayesian Filter: the *prediction* step and the *update* step. Assuming Gaussian distributions for the factors in Equation 4.6 with linear relationships between the random variables, the standard *Kalman Filter* (KF) is derived.

To compute the probability of the hidden state given the observation, as in Equation 4.3, one step is still missing: to compute the integral in the denominator (compute the marginal probability $P(y)$). The computation of the integral is often intractable, even for simple cases. Therefore, methods that approximate the probability $P(x|y)$ (approximate inference methods) have been developed.

Approximate methods output a probability distribution $Q(x)$ that most accurately resembles the original $P(x|y)$. To this end, a measure of similarity between probability distributions is required. Information theory provides such a measure, commonly known as *Kullback-Leibler divergence* (KL) [170]:

$$KL(Q||P) = \int_{-\infty}^{+\infty} Q(x) ln \frac{Q(x)}{P(x)} dx \tag{4.7}$$

KL is not a proper distance metric, as it is not symmetrical; namely, $KL(Q||P) \neq KL(P||Q)$. However, it can be used as a quasi-metric; since it is non-negative and is zero only when P and Q coincide. Therefore, a reasonable approximation $Q(x)$ of $P(x|y)$ could be obtained by minimising the KL-divergence between the distributions P and Q:

$$Q^*(x) = \arg\min_{Q \in \mathbf{Q}} KL(Q(x)||P(x|y)) \tag{4.8}$$

where Q^* denotes the optimal approximate distribution and $\mathbf{Q}$ is the space of all candidate approximate distributions. Solving for $Q^*(x)$ yields the recursive formulae for a Bayesian Filter.

The optimisation problem, as described in Equation 4.8, can be solved using *Variational Bayes Approximation*. This approximate inference method is characterised as variational due to the similarity to the Calculus of Variations. In Calculus of Variations, the goal is to find a function $f(x)$ that minimises a specific *functional* $F[f(x)]$; which maps a function to a real number. Similarly, KL-divergence is a functional of probability distributions; that, in turn, are functions. Hence, Variational Bayes Approximation attempts to find the probability distribution $Q(x)$ (function of the hidden variable x), so as to minimise the functional $KL(Q||P)$. To this end, the *complete data log-likelihood*, defined as $\langle logp(x_{1:k}, y_{1:k}) \rangle$, is required; where the brackets indicate expected values and *log* is the natural logarithm[1]. The joint probability distribution $logp(x_{1:k}, y_{1:k})$ can be factorised using a graphical model, as described previously.

[1] Any base can be used for the logarithm; the important thing is to remain consistent throughout the filter derivation

The steps for devising an arbitrary Bayesian filter are the following:

1. Choose the random variables of the system to be modelled
2. Define the conditional dependencies between the random variables using a Dynamic Bayesian Network
3. Using the DBN, decompose the joint probability of all the random variables to a product of factors
4. Define a probability distribution for each of the factors
5. Apply Variational Bayes Approach to derive the filtering equations.

Variational Bayes Approximation provides straightforward solutions to the optimization problem depicted in 4.8. Nevertheless, a detailed description of the theoretical framework is beyond the scope of this chapter. A simple, albeit rigorous, outline of the theory can be found in Appendix C.

A bayesian filter that is robust to sensor outliers requires that the measurement covariance is treated as hidden variable as well; i.e., it is computed based on incoming information. A graphical model that captures this requirement is illustrated in Figure 4.4.

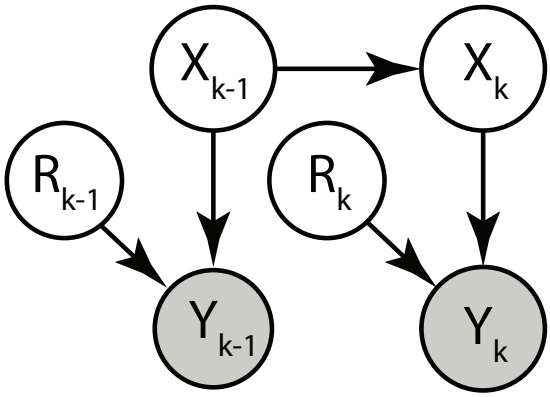


Figure 4.4: Kalman Filter with variable measurement noise model used for outlier robust filtering

In the remaining of the chapter, two bayesian filters are presented in increasing order of complexity. Specifically, steps 1-3 are identical for both filters. However, the probability distributions that model each of the factors (i.e., step 4) are different; mainly, in the way the covariance of the measurement model is defined.

4.2 A Simple Outlier Robust Filter

In this section, the outlined theory is employed to develop a non-linear filter that is robust to sensor degradation. Complying with the standard Bayesian Filter formulation, the algorithm requires a transition and a measurement model. The transition model, namely $P(\mathbf{x}_{k+1}|\mathbf{x}_k)$, provides the probability distribution of the state $\mathbf{x} \in \mathbb{R}^{d_1}$ at time $k+1$, given the state at k. In turn, the measurement model $P(\mathbf{y}_k|\mathbf{x}_k)$represents the probability distribution of the sensor output $\mathbf{y}_k \in \mathbb{R}^{d_2}$ given the state. This filter is an extension the work presented in Ting et al. [1], where the authors present an outlier robust filter for zero-mean stationary processes. In this section, Ting's work is extended for arbitrarily complex non-linear processes. As previously mentioned, an outlier robust filter affords a varying measurement covariance. Ting et al. make the following assumption for the measurement covariance $\mathbf{R}_k$ at time instant t:

$$\mathbf{R}_k = 1/w_k \mathbf{R}_{init} \tag{4.9}$$

where $w_k \in \mathbb{R}$ and $\mathbf{R}_{init} \in \mathbb{R}^{d_2 \times d_2}$ is an initial estimation of the measurement covariance. The value of w_k is a hidden variable; i.e., the value of w_k is computed in real-time based on incoming information. Specifically, in the presence of outliers w_k converges to small values. Low values of w_k increase the resulting measurement covariance; thus, indicating reliance more on the model rather than the sensor measurement. The weight w_k is sampled from a Gamma distribution with shape parameters $\alpha_{w_k}, \beta_{w_k}$. In Bayesian inference, such a distribution is known as the *prior* distribution and is used to incorporate expert knowledge about the sampled hidden variable. Gamma distribution is defined on $(0, \infty)$; thereby, $w_k \neq 0$ for any combination of the parameters $\alpha_{w_k}, \beta_{w_k}$.

4.2.1 Filter Derivation

The filter that employs the measurement covariance as described by Equation (4.9) is illustrated in Figure 4.5; where, the hidden measurement covariance $\mathbf{R}_k$, as appears in the general filter representation, has been substituted by w_k. This is due to the fact that $\mathbf{R}_{init}$ is constant.

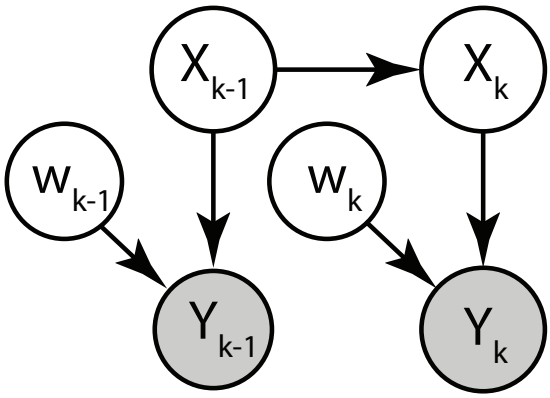


Figure 4.5: Kalman Filter with varying measurement covariance; the resulting covariance is equal to the initial, scaled by a weight. The weight is computed at runtime [1]

Based on the Bayesian Network of Figure 4.5, the joint probability distribution of all the variables at all times factorises as follows:

$$P(\mathbf{x}_{1:k}, \mathbf{y}_{1:k}, w_{1:k}) = \prod_{i=1}^{k} P(\mathbf{y}_i|\mathbf{x}_i) \prod_{i=1}^{k} P(\mathbf{x}_i|\mathbf{x}_{i-1}) P(\mathbf{x}_0) \tag{4.10}$$

Taking the natural logarithm of the above expression, the *complete data log-likelihood* $lnp(\mathbf{x}_{1:k}, \mathbf{y}_{1:k}, w_{1:k})$ is computed:

$$\begin{aligned} lnp(\mathbf{x}_{1:k}, \mathbf{y}_{1:k}, w_{1:k}) &= \sum_{i=1}^{k} lnp(\mathbf{y}_i|\mathbf{x}_i, w_i) \\ + \sum_{i=1}^{k} lnp(\mathbf{x}_i|\mathbf{x}_{i-1}) &+ \sum_{i=1}^{k} lnp(w_i) + p(\mathbf{x}_0) \end{aligned} \tag{4.11}$$

Having factorised the joint probability distribution according to the Bayesian Network at hand, next all the yielded factors need to be described:

$$P(\mathbf{x}_{k+1}|\mathbf{x}_k) \sim \mathcal{N}(f(\mathbf{x}_k, \mathbf{u}_k), \mathbf{Q}) \tag{4.12}$$

$$P(\mathbf{y}_k|\mathbf{x}_k, w_k) \sim \mathcal{N}(\mathbf{C}\mathbf{x}_k, w_k^{-1}\mathbf{R}_{init}) \tag{4.13}$$

$$P(w_k) \sim Gamma(\alpha_{w_k}, \beta_{w_k}) \tag{4.14}$$

where $\mathbf{Q} \in \mathbb{R}^{d_1 \times d_1}$ is the covariance of the transision model (dynamic model), $\mathbf{R}_{init} \in \mathbb{R}^{d_2 \times d_2}$ the covariance of the measurement model, and $\mathbf{C} \in \mathbb{R}^{d_2 \times d_1}$ the output measurement matrix; assuming a linear measurement model as in the standard Kalman filter. Ting et al. [1] focus on first-order autoregressive stationary (or slowly drifting) processes. Here, the algorithm is extended to a more general class of systems; specifically, nonlinear systems with external inputs. To this end, the transition model is modified to include a nonlinear map $f(\mathbf{x}_k, \mathbf{u}_k) : \mathbb{R}^{d_1} \times \mathbb{R}^{d_3} \mapsto \mathbb{R}^{d_1}$, which models the dynamics of the system as a function of the previous state $\mathbf{x}_k$ and the control input $\mathbf{u}_k \in \mathbb{R}^{d_3}$. Conversely, the measurement model remains as appears in [1].

Substituting (4.12), (4.32), (4.14) into (4.11) yields:

$$\begin{aligned}
lnp(\mathbf{x}_{1:k}, \mathbf{y}_{1:k}, w_{1:k}) &= \frac{d_2 - 2}{2} \sum_{i=1}^{k} lnw_i - \frac{k}{2} ln|R| \\
&- \frac{1}{2} \sum_{i=1}^{k} w_i (\mathbf{y}_i - \mathbf{C}\mathbf{x}_i)^T \mathbf{R}^{-1} (\mathbf{y}_i - \mathbf{C}\mathbf{x}_i) - \frac{k}{2} ln|\mathbf{Q}| \\
&- \frac{1}{2} \sum_{i=1}^{k} (\mathbf{x}_i - f(\mathbf{x}_{i-1}, \mathbf{u}_{i-1}))^T \mathbf{Q}^{-1} (\mathbf{x}_i - f(\mathbf{x}_{i-1}, \mathbf{u}_{i-1})) \\
&+ \sum_{i=1}^{k} a_{w_i} lnw_i - \sum_{i=1}^{k} b_{w_i} w_i \\
&+ \sum_{i=1}^{k} (a_{w_i} lnb_{w_i} - ln\Gamma(a_{w_i})) + \text{constant}
\end{aligned} \tag{4.15}$$

It is worth noting that the leading terms in equation (4.15) are independent of $\mathbf{x}_k$ and w_k; hence, they could be absorbed within the constant term. However, they depend on the parameters α and β of the Gamma distribution. To this end, they are essential for deriving an analytical expression for the parameters of the Gamma

distribution.

Similarly to [1], applying the following mean-field approximation (see Equation (4.16)) and using the result of equation (C.5),the optimal approximate distribution $\widetilde{P}^*(w,x)$of the hidden variables are derived.

$$\widetilde{P}(\mathbf{x}_{1:k}, w_{1:k}) = \prod_{i=1}^{k} \widetilde{P}(w_i) \prod_{i=1}^{k} \widetilde{P}(\mathbf{x}_i|\mathbf{x}_{i-1}) \widetilde{P}(\mathbf{x}_0) \tag{4.16}$$

Specifically, the optimal approximate distribution of the state $\widetilde{P}^*(\mathbf{x}_k|\mathbf{x}_{k-1})$ is normally distributed with mean $\langle \mathbf{x}_k \rangle$ and variance $\mathbf{\Sigma}_k$. On the other hand, w_k follows a Gamma distribution with parameters $\alpha_{opt} = \frac{d_2}{2} + \alpha_{w_k}$ and $\beta_{opt} = \beta_{w_k} + \frac{1}{2}\langle (\mathbf{y}_k - \mathbf{C}\mathbf{x}_k)^T \mathbf{R}^{-1} (\mathbf{y}_k - \mathbf{C}\mathbf{x}_k) \rangle$. $\widetilde{P}(\mathbf{x}_0)$ is the prior distribution of the state, which is constant; hence, it does not influence the derivation of the filter equations. The analytical expressions for $\langle \mathbf{x}_k \rangle$ and variance $\mathbf{\Sigma}_k$ are given by Equations (4.18) and (4.17) respectively. Moreover, given the parameters α_{opt} and β_{opt} the expected value of the hidden variable w_k can be computed as $\frac{\alpha_{opt}}{\beta_{opt}}$ (see Equation (4.19)).

$$\mathbf{\Sigma}_k = (\langle w_k \rangle \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C} + \mathbf{Q}^{-1})^{-1} \tag{4.17}$$

$$\langle \mathbf{x}_k \rangle = \mathbf{\Sigma}(\mathbf{Q}^{-1} \langle f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \rangle + \langle w_k \rangle \mathbf{C}^T \mathbf{R}^{-1} \mathbf{y}_k) \tag{4.18}$$

$$\langle w_k \rangle = \frac{\frac{d_2}{2} + \alpha_{w,k}}{\beta_{w,k} + \frac{1}{2}\langle (\mathbf{y}_k - \mathbf{C}\mathbf{x}_k)^T \mathbf{R}^{-1} (\mathbf{y}_k - \mathbf{C}\mathbf{x}_k) \rangle} \tag{4.19}$$

The only term that needs further manipulation is that of $\langle f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \rangle$, namely the expected value of the nonlinear dynamic function. To compute $\langle f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \rangle$, a Taylor expansion of the function $f(\mathbf{x}_k, \mathbf{u}_k)$ around the previous state-input point (x_{k-1}, u_{k-1}) has been utilised. By discarding the second order terms and taking the expectation upon the partial series expansion, we end up with the following recursive formula for the indicated expected value:

$$\begin{aligned} \langle f(\mathbf{x}_k, \mathbf{u}_k) \rangle = \langle f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \rangle + \mathbf{J}_x [\langle \mathbf{x}_k \rangle - \langle \mathbf{x}_{k-1} \rangle] \\ + \mathbf{J}_u [\mathbf{u}_k - \mathbf{u}_{k-1}], k > 1 \end{aligned} \tag{4.20}$$

where $\mathbf{J}_x$ and $\mathbf{J}_u$ are the Jacobians of $f(\mathbf{x}_k, \mathbf{u}_k)$ with respect to $\mathbf{x}$ and $\mathbf{u}$, evaluated

at $(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$. To initialise the recursive relation described in Equation (4.20), it was assumed that $\langle f(\mathbf{x}_0, \mathbf{u}_0) \rangle \approx f(\mathbf{x}_0, \mathbf{u}_0)$.

After deriving an analytical expression for the approximate distribution $\widetilde{P}$, a way to tune the parameters of the Gamma distribution is required. Setting the partial derivatives of $lnp(\mathbf{x}_{1:k}, \mathbf{y}_{1:k}, w_{1:k})$ with respect to α and β equal to zero, yields the respective update equations:

$$ln\alpha_{w_k} - ln\overline{w} + \overline{lnw} - \psi(\alpha_{w_k}) = 0 \tag{4.21}$$

$$\beta_{w_k} = \frac{\alpha_{w_k}}{\overline{w}} \tag{4.22}$$

where $\overline{w}$ indicates the arithmetic mean of w, and $\psi(a)$ is the digamma function [171].

4.3 A synthetic example

In this section, the proposed algorithm is evaluated in simulation. A non-linear process (Equation (4.23)) is simulated to generate the state sequence. The measurements consist of the state sequence after injecting some white noise. The goal of this section is to monitor the behaviour of the filter, as the measurement is either corrupted by outliers or frozen at a constant value.

$$x(t) = e^{-0.01t} cos(2\pi 0.003t) \tag{4.23}$$

By taking the derivative with respect to time t and then discretising, the dynamics can be written in the form: $x_{k+1} = f(x_k, u_k)$. Next, the nonlinear mapping is learnt using LWPR. LWPR was tuned using the standard procedure, as described in Section 3.3.2. As long as the Jacobians $\mathbf{J}_x$ and $\mathbf{J}_u$ can be computed, the choice of modelling method for the dynamics does not influence the performance of the filter. LWPR's interface provides direct access to both the indicated Jacobians.

Figure 4.6 depicts the performance of the proposed algorithm, compared to the

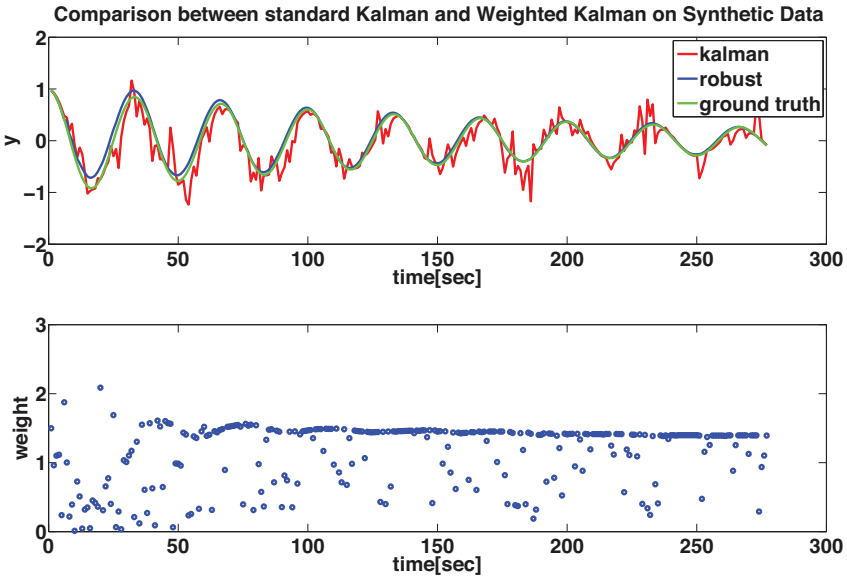


Figure 4.6: The top figure compares the performance of the algorithm with the standard Kalman filter for the simulated system, as described in equation (4.23). In this experiment, the measurements were corrupted with outliers. The bottom figure shows the sampled weight in each iteration of the algorithm

standard kaman filter (top). In this simulation, the measurement sequence was corrupted with outliers. The probability of an outlier generation was set to 0.4. It is obvious that the presented filter outperforms the standard kaman filter. Also, the weight w_k approaches zero in the presence of an outlier. As a result the measurement covariance tends to infinity (in practice to a large value); shifting the responsibility of the state estimation to the model.

Figure 4.7 shows the output of the filter compared to the ground truth when the sensor got stuck to a fixed value. Also, in this case, the filter reconstructs the state of the system quite accurately. Apparently, modelling errors accumulate over time; leading to a slow drift of the model estimation far from the real value. However, the prediction remains accurate sufficiently long (several seconds in this case) to enable appropriate action from the decision-making mechanisms of the agent. Figure 4.8 illustrates the prior probability distribution of w_k. The indicated distribution was plotted using the parameters α_{w_k}, β_{w_k} as computed by Equation (4.21). The resulting distribution assigns almost infinite probability to $w_k \to 0$. This is due to the constant misalignment between the sensor value and the model prediction.

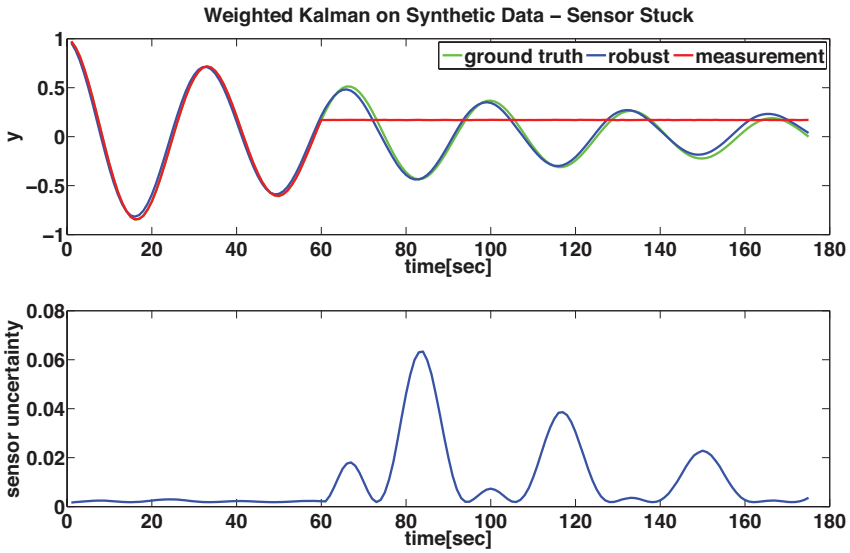


Figure 4.7: The top figure illustrates the output of the algorithm for the simulated system, as described in equation (4.23). In this experiment, the measurement was stuck to a constant value of 0.15. The bottom graph shows the effective measurement uncertainty (i.e., R/w_k) throughout the simulation.

4.3.1 Experimental Results

In this section, the proposed algorithm is tested on real-world data. The following sensor failures have been considered: 1) the sensor is stuck at a fixed value and 2) the sensor measurement is corrupted by outliers. The first failure mode is fairly easy to simulate. For the second case, a random number p is sampled from a standard uniform distribution. If $p \leq \epsilon$ then an offset is added to the measurement. The offset itself is sampled from a normal distribution with zero mean and variance $\lambda \gg \sigma_{sensor}$. In this way, each measurement is corrupted with outliers with probability ϵ.

In the following experiments, the case of a defective DVL has been considered. The DVL measures the linear velocity of the vehicle. We want to show that in the event of a simulated DVL failure, the algorithm utilises the dynamic model to estimate the velocity of the vehicle correctly. To this end, the original DVL measurement for the vehicle's surge velocity is considered as ground truth. Following that, the surge measurement is either corrupted with outliers or frozen to a constant value. The resulting measurement sequence (i.e., with the corrupted data) is given to the algorithm. The latter estimates the surge velocity of the vehicle given the corrupted measurements.

Figure 4.9 compares the velocity profiles, as computed by the standard Kalman

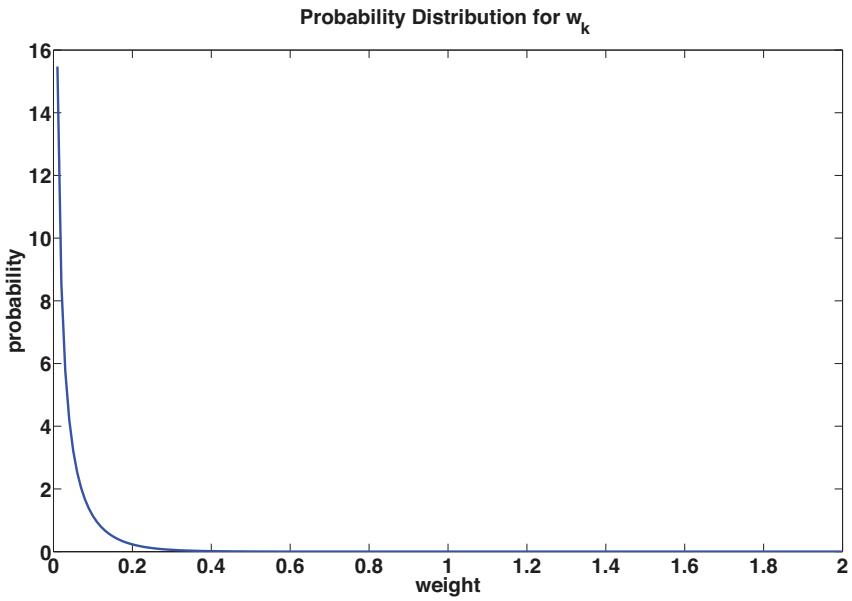


Figure 4.8: Probability distribution $P(w_k) \sim Gamma(a, b)$. The computed shape and scale parameters for the distribution are such that the probability of small values for w_k tends infinity. Intuitively, the algorithm does not rely on the sensor reading and favours the model prediction. This is a result of the constant mismatch between the sensor and the model, which followed the simulated sensor failure.

Filter and the proposed filtering algorithm. In this experiment, the original measurement from the DVL was corrupted with outliers. The outliers were generated with probability $\epsilon = 0.25$ ($\lambda = 1$). The filter was initialised with $\mathbf{Q} = \mathbf{R} = 0.001\mathbf{I}^{3\times3}, \mathbf{C} = \mathbf{I}^{3\times3}$, where $\mathbf{I}^{3\times3}$ is the 3×3 identity matrix. The corrupted measurement is illustrated in the bottom subfigure. Evidently, the performance of the algorithm is not influenced by the presence of outliers. Likewise, Figure 4.10 compares the estimated velocity of the vehicles when the DVL output froze at a specific value. Again the suggested algorithm was able to compensate for the failed sensor and computed the velocity accurately.

The filter described in this section performs as required. Nevertheless, w_k penalises all the measurement dimensions equally. In certain occasions, this may prove to be problematic. When applied to the robot, this issue was resolved by using three filters in parallel; i.e., one for each dimension: *surge*, *sway*, and *yaw rate*. This is an acceptable remedy for lower dimensional problems; albeit a more general filter would be desirable. For this reason, a more general filter is derived, by relaxing the assumptions, concerning the measurement covariance, which were made in this section.

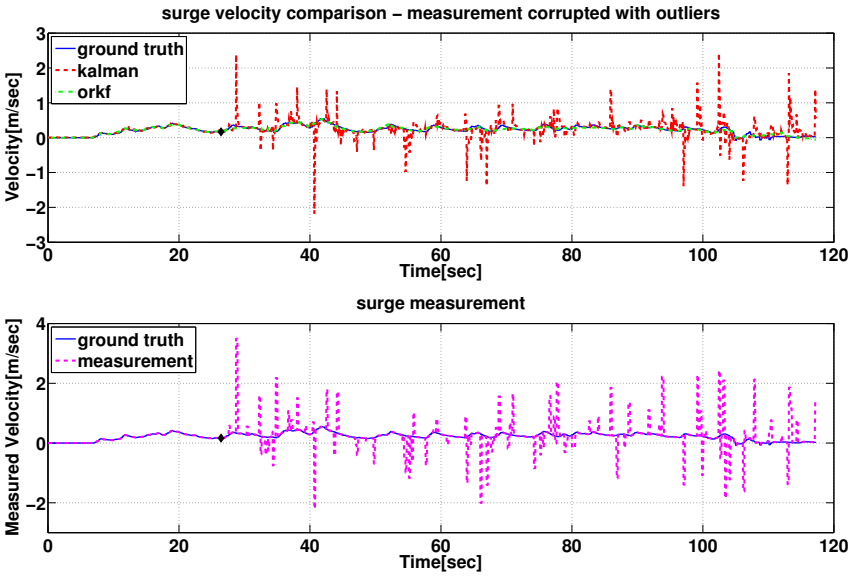


Figure 4.9: The figure on the top compares the estimated surge velocity, as computed by the standard Kalman filter and the suggested algorithm. At the bottom, the outlier corrupted measurement which is considered by both filters is shown. The time instant at which the failure occurred is marked with a black diamond.

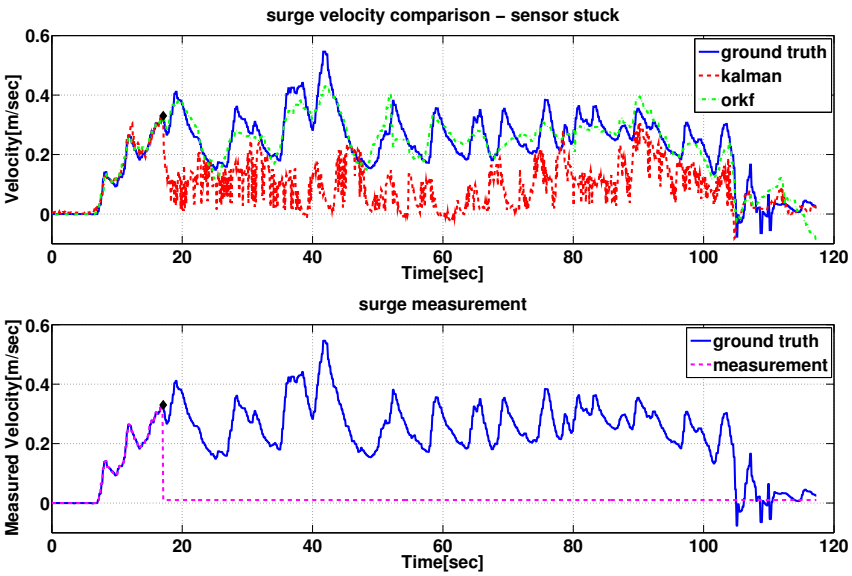


Figure 4.10: The figure on the top compares the surge velocities when the measurement is stuck at a constant value equal to 0.01. At the bottom, the output of the "stuck" sensor is plotted against the original measurement.

4.4 Self-Tuning Kalman Filter

The Self-Tuning Kalman Filter (STKF) is an extension of the of the Outlier Robust Kalman Filter for non-stationary processes with non-linear systems. Similarly to the previous algorithm, STKF is developed using the Variational Bayes Approximation; the difference lies in the assumptions of the measurement noise model. Specifically, STKF samples the measurement covariance from the general class of covariance matrices [152]; whereas, the previous algorithm assumes diagonal covariances that scale homogeneously in all dimensions.

The non-linear dynamics are modelled using LWPR. Moreover, STKF utilises LWPR's confidence interval prediction to update the process noise covariance matrix. In this way, both the process and measurement covariance are updated online; yielding a recursive estimation algorithm that can tune the required parameters autonomously. STKF will be showcased on the problem of underwater navigation; moreover, STKF's robustness to sensor failures will be demonstrated.

The purpose of STKF is to estimate the vehicle's instantaneous velocity with respect to the body frame; i.e., a coordinate frame attached to the robot (see Figure 4.11). Next, the velocity is converted from the body frame to the global frame using the robot's kinematic equations. The velocities in the global frame are integrated appropriately to yield the pose of the robot. The major advantage of the indicated algorithm is the inherent robustness to sensor failures. The algorithm discards measurements from defective sensors as outliers. Apart from training the dynamic model, the navigation system requires no further tuning.

Two variants of the KF were combined to yield the filter discussed above. Specifically, the *prediction* step is that of the Extended Kalman Filter (EKF). The prediction step uses a dynamic model to compute the velocity of the vehicle, given the input to the thrusters. Next, the *measurement* step of the Outlier Robust Kalman Filter (ORKF) corrects the model prediction, using the information from the sensors. In this section, both the prediction and the update step of the filter are discussed. Furthermore, the theoretical background for the complete navigation framework is provided.

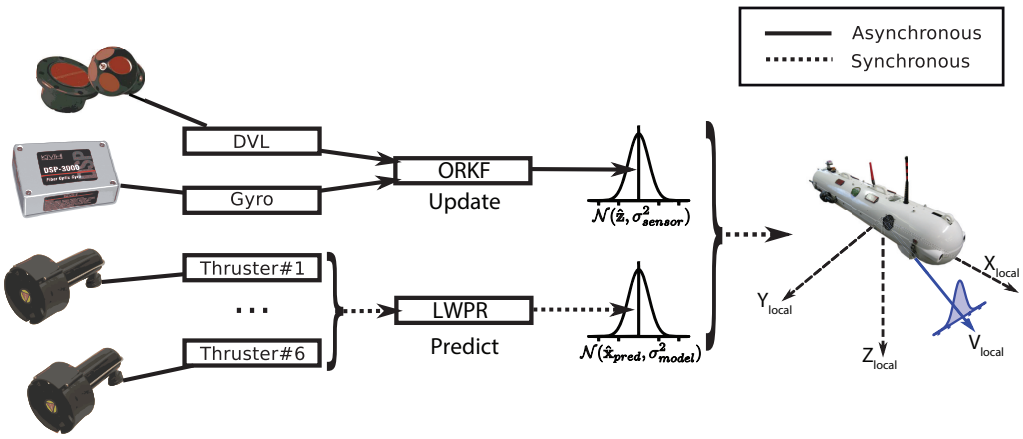


Figure 4.11: Schematic representation of the Self-Tuning Kalman Filter (STKF) that estimates the instantaneous velocity of the vehicle in the body frame. During the *prediction* step, the dynamic model, based on Locally Weighted Projection Regression, estimates the robot's state (i.e., the surge velocity, the sway velocity and the yaw rate). The *prediction* step is executed synchronously at a frequency of 30 Hertz. The predicted state $\hat{\mathbf{x}}_{pred}$ and the respective confidence interval σ_{model} define a probability distribution $\mathcal{N}(\hat{\mathbf{x}}_{pred}, \sigma^2_{model})$ for the state. Conversely, the *update* step occurs asynchronously; whenever a new measurement $\mathbf{z}_k$ arrives. During the *update* step, the algorithm estimates the variance of the measurement σ_{sensor}, as in the Outlier Robust Kalman Filter; yielding a probability distribution $\mathcal{N}(\mathbf{z}_k, \sigma^2_{sensor})$ for the measurement. The resulting probability distributions are combined as in the standard Kalman Filter, to compute the instantaneous velocity of the vehicle in the body frame

4.4.1 Prediction step

One of STKF's differences is that it uses machine learning, and in particular kernel regression [172], to model the vehicle's dynamics. As a kernel-based algorithm, LWPR is *non-parametric*; i.e., it abstractly models the relation between an independent and a dependent (target) variable, by exploiting previously seen data. Similarly to EKF, the prediction step is defined as follows:

$$\mathbf{x}_{k+1|k} = f(\mathbf{x}_{k|k}, \mathbf{u}_k) \tag{4.24}$$

$$\mathbf{P}_{k+1|k} = \mathbf{F}_k \mathbf{P}_{k|k} \mathbf{F}_k^T + \mathbf{Q}_k \tag{4.25}$$

where $\mathbf{x}_{k|k}$ is the state estimation at time k. In this application, the state comprises the *surge* and *sway* velocity, as well as the *yaw rate*. The input vector $\mathbf{u}_k$ consists of the throttle (i.e., percentage of available thrust), applied to each of the thrusters. $\mathbf{Q}_k$ is the *process (model) covariance*. The process covariance quantifies the model's

prediction accuracy, whereas $\mathbf{P}_k$ is the covariance of the final state estimation. $\mathbf{F}_k$ is the *Jacobian* of the dynamic model (see Equation (4.26)) computed at a specific point in the state-input space.

$$\mathbf{F}_k = \left.\frac{\partial f(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x}_k}\right|_{(\mathbf{x}_0, \mathbf{u}_0)} \tag{4.26}$$

The Jacobian is used to linearise the dynamic equation around $(\mathbf{x}_0, \mathbf{u}_0)$. STKF uses the confidence interval of LWPR predictions (section A.3.2) to update the process covariance $\mathbf{Q}_k$ during runtime. In this application of the filter, the dimensions of the state are assumed to be mutually independent. Therefore, $\mathbf{Q}_k$ is a diagonal matrix. The algorithm adjusts $\mathbf{Q}_k$ in the following manner:

$$\mathbf{Q}_k = \text{diag}\{\sigma_{pred}^{2,(1)}, ..., \sigma_{pred}^{2,(d)}\} \tag{4.27}$$

where $\sigma_{pred}^{(i)}$ is the computed confidence interval for the i-th state dimension and d is the dimension of the state. Given the dynamic model $\mathbf{x}_{k+1|k} = f(\mathbf{x}_{k|k}, \mathbf{u}_k)$, the process noise covariance $\mathbf{Q}_k$, and the Jacobian $\mathbf{F}_k$, the *prediction* step is fully defined.

4.4.2 The update step

Similarly to KF, the update step of STKF corrects the state prediction based on new sensor information. The model and measurement accuracy is what defines the contribution of each to the final estimation. This information is encoded in the respective covariance matrix; i.e., $\mathbf{Q}_k$ for the dynamic model and $\mathbf{R}_k$ for the sensors. The distinguishing characteristic of STKF is the variable measurement covariance. At every time instant, $\mathbf{R}_k$ is computed in a way that maximises the likelihood of the *filter innovation*; i.e., the difference between the model prediction and the measurement. Intuitively, STKF identifies the most probable explanation, in terms of $\mathbf{R}_k$, for the observed discrepancy between the model and the sensors. The update

step is formulated as follows:

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{C}_k \cdot \mathbf{x}_{k+1|k} \tag{4.28}$$

$$\mathbf{H}_k = \mathbf{C}_k \cdot \mathbf{P}_{k+1|k} \cdot \mathbf{C}_k^T + \mathbf{R}_k \tag{4.29}$$

where $\tilde{\mathbf{y}}_k$ is the filter innovation (residual). The output matrix $\mathbf{C}_k$ selects the part of the state that is relevant to each sensor. The estimated covariance of the filter innovation $\mathbf{H}_k$ is used to compute the Kalman gain $\mathbf{K}_k$; which acts as the weighting factor between the model prediction and the measurements. The smaller the values in $\mathbf{R}_k$, the more trustworthy are the measurements regarded by the filter. In the presence of a sensor failure, the erratic behaviour needs to be reflected onto the sensor's measurement covariance. In this case, $\mathbf{R}_k$ should increase; hence, pushing all the computation responsibility to the model. To this end, the *update* step of the Outlier Robust Kalman Filter [152] has been utilised.

ORKF samples the inverse of the measurement covariance (measurement precision) in each iteration from a Wishart distribution [173]. The Wishart distribution extends the gamma distribution for non-integer degrees of freedom. Moreover, it is the conjugate prior of a precision matrix $\mathbf{S}^{-1}$; i.e., the inverse covariance of a multivariate normally distributed variable $\mathbf{x}$ with known mean. Therefore, the Wishart distribution is the proper choice for sampling the measurement precision matrix. The Wishart distribution is parametrised as follows:

$$\mathbf{S^{-1}} \sim \mathcal{W}(\mathbf{\Lambda}, s) \tag{4.30}$$

where $\mathbf{\Lambda}$ is the seed matrix and s is the degrees of freedom of the distribution. The mean of the distribution is equal to $\mathbf{m} = s\mathbf{\Lambda}$. Hence, a reasonable choice for $\mathbf{\Lambda}$ is $\mathbf{R}_{init}^{-1}/s$. $\mathbf{R}_{init}$ is an initial covariance matrix and is defined using the respective accuracy properties of the sensors. The matrix $\mathbf{S^{-1}}$ is invertible with probability one, as long as the seed matrix $\mathbf{\Lambda}$ is invertible. s controls how concentrated the

distribution is around its mean. For high values of s, the probability density function of a Wishart distribution converges to that of a normal distribution. Lower values of s yield distributions with thicker tails. In this way, measurements far from the mean are assigned a non-negligible probability. The inverse covariance is sampled until the log-likelihood of the innovation is maximum (see Equation (4.31)). After convergence, the resulting measurement covariance $\mathbf{R}_k$ is given by Equation (4.32):

$$\mathcal{L}(\tilde{\mathbf{y}}_k) = -\frac{1}{2} log|\mathbf{H}_k| - \frac{1}{2}\tilde{\mathbf{y}}_k^T \mathbf{H}_k^{-1}\tilde{\mathbf{y}}_k + constant \tag{4.31}$$

$$\mathbf{R}_k = \frac{s_k \mathbf{R}_{init} + \mathbf{S}_k}{s_k + 1} \tag{4.32}$$

where:

$$\mathbf{S}_k = \langle \tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k^T \rangle + \mathbf{P}_{k|k} \tag{4.33}$$

The notation $\langle x \rangle$ indicates the expected value of x. From equation (4.32), it is implied that for $s_k \to \infty$ the resulting measurement covariance coincides with $\mathbf{R}_{init}$. In this case, the filter behaves exactly like the standard Kalman Filter. Conversely, small values of s_k give more weight to the sampled measurement covariance $\mathbf{S}_k$. The degrees of freedom s are computed by equation (4.34). This equation has a unique solution, and is found in each iteration numerically.

$$\frac{1}{n}\sum_{t=1}^{k} log|\mathbf{R}_t|^{-1} - log\left|\frac{1}{k}\sum_{t=1}^{k}\mathbf{R}_t^{-1}\right| + \sum_{i=1}^{d}\frac{s}{2} - \psi\left(\frac{s+1-i}{2}\right) - \sum_{i=1}^{d}\frac{1}{k}\sum_{t=1}^{k}\frac{\nu_t}{2} - \psi\left(\frac{\nu_t+1-i}{2}\right) = 0 \tag{4.34}$$

where $\nu_t = s_t + 1$, and ψ is the *digamma* function [171]. As seen from Equation (4.34) the computation involves the history of the measurement covariance $\mathbf{R}_k$. In this way, the degrees of freedom takes into account previous experience as a bias. Specifically, in an experiment with many outliers, terms that involve sums of $\mathbf{R}_k$ will gradually increase. This is due to the fact that the algorithm will use higher variances to compensate for the outliers. The larger the indicated sums, the smaller the yielded degree of freedom s. In other words, previous experience indicates that the measurements are not to be trusted.

Given Equations (4.32), (4.33), and (4.34) the measurement covariance $\mathbf{R}_k$ is fully defined. It is worth noting that the indicated equations are *coupled*; i.e., to compute $\mathbf{R}_k$, the value for s_k is required and vice versa. A typical solution to this problem is provided by *Expectation-Maximization* (EM) algorithm [174]. Starting from an initial value for both $\mathbf{R}_k$ and s_k, Equations (4.32), (4.33), and (4.34) are utilised to cyclically update the respective values until convergence. The values from the previous iteration constitute a very good initial point for the iterative procedure. For the very first iteration, it is advised by the authors of ORKF to use $s = d - 1$ where d is the dimension of the state; and $\mathbf{R}_{init}$, as indicated by the sensor accuracy specifications. In practice, EM requires no more than five iterations to converge. Following convergence, the remaining steps of the standard Kalman

Filter are executed as follows:

$$\mathbf{K}_k = \mathbf{P}_{k+1|k}\mathbf{C}_k^T\mathbf{H}_k^{-1} \tag{4.35}$$

$$\mathbf{x}_{k+1|k+1} = \mathbf{x}_{k+1|k} + \mathbf{K}_k\tilde{\mathbf{y}}_k \tag{4.36}$$

$$\mathbf{P}_{k+1|k+1} = (\mathbf{I} - \mathbf{K}_k\mathbf{C}_k)\mathbf{P}_{k+1|k} \tag{4.37}$$

4.4.3 STKF summary

STKF adjusts all the required parameters automatically. The model covariance $\mathbf{Q}_k$ is computed as described in Section A.3.2. Following that, the measurement covariance $\mathbf{R}_k$ is chosen at runtime (see Section 4.4.2). Thus, the filter tunes both the process and measurement covariance during execution. Since properly adjusting the covariances is often very cumbersome, the self-tuning feature is an important advantage of the algorithm. Algorithm 1 summarises STKF.

Algorithm 1 Self-Tuning Kalman Filter

$\mathbf{x}_k, \mathbf{P}_k, \mathbf{z}_k, \mathbf{R}_{init}$
procedure PREDICTION STEP
    $\mathbf{x}_{k+1|k}, \sigma^2_{pred}, \mathbf{F}_k \leftarrow \text{LWPR}(\mathbf{x}_k, \mathbf{u}_k)$
    $\mathbf{Q}_k \leftarrow \text{diag}\{\sigma^2_{pred}\}$
    $\mathbf{P}_{k+1|k} \leftarrow \mathbf{F}_k\mathbf{P}_k\mathbf{F}_k^T + \mathbf{Q}_k$
end procedure
procedure UPDATE STEP
    $\tilde{\mathbf{y}}_k \leftarrow \mathbf{z}_k - \mathbf{C}_k \cdot \mathbf{x}_{k+1|k}$
    for until $\mathcal{L}(\tilde{\mathbf{y}}_k)$ convergence **do**
        $\mathbf{S}_k = \langle\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k^T\rangle + \mathbf{P}_k$
        $s \leftarrow$ COMPUTE_DOFS()
        $\mathbf{R}_k \leftarrow (s\mathbf{R}_{init} + \mathbf{S}_k)(s+1)^{-1}$
        $\mathbf{H}_k \leftarrow \mathbf{C}_k \cdot \mathbf{P}_{k+1|k} \cdot \mathbf{C}_k^T + \mathbf{R}_k$
        $\mathbf{K}_k \leftarrow \mathbf{P}_{k+1|k}\mathbf{C}_k^T\mathbf{H}_k^{-1}$
        $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_{k+1|k} + \mathbf{K}_k\tilde{\mathbf{y}}_k$
        $\mathbf{P}_{k+1} \leftarrow (\mathbf{I} - \mathbf{K}_k\mathbf{C}_k)\mathbf{P}_{k+1|k}$
    end for
end procedure

function COMPUTE_DOFS()
    Numerical Solver for Equation (4.34)
end function

The need for model training may seem like a drawback at first. However, training in that case is the equivalent of an identification experiment that is needed to choose

the parameters of a parametric (e.g., state space) model. Training the dynamic model using the LWPR framework is straightforward (see Section 3.3.2). Besides, LWPR's adaptivity affords the fast recalibration of an existing core model. In this way, variations in the vehicle's configuration and operating conditions are treated effectively.

4.4.4 Velocities in the world coordinate frame

The discussed filter estimates the vehicle's instantaneous velocity in the body frame. However, the goals of a mission are generally given in the world coordinate frame. To convert the robot's velocity between the two coordinate frames, the velocity Jacobian is utilized. A detailed derivation of the Jacobian is beyond the scope of this thesis. It is important to note that the velocity Jacobian depends on the representation for the orientation. Specifically, for the case of Roll-Pitch-Yaw angles (ϕ, θ, ψ) the Jacobian is as follows [139]:

$$\mathbf{J} = \begin{bmatrix} \mathbf{R}_I^B & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{J}_o \end{bmatrix} \tag{4.38}$$

Where:

$$\mathbf{R}_I^B = \begin{bmatrix} c_\psi c_\theta & s_\psi c_\theta & -s_\theta \\ -s_\psi c_\phi + c_\psi s_\theta s_\phi & c_\psi c_\phi + s_\psi s_\theta s_\phi & s_\phi c_\theta \\ s_\psi s_\phi + c_\psi s_\theta c_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi & c_\phi c_\theta \end{bmatrix}$$

$$\mathbf{J}_o = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix}$$

Assuming small values for roll (ϕ) and pitch (θ), the aforementioned matrices are simplified to:

$$\mathbf{R}_I^B = \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{J}_o = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

During the experiments, the roll and the pitch remained constant to zero. The current infrastructure does not control the indicated degrees of freedom. Furthermore, the robot was driven around smoothly, to avoid an accidental excitation of the roll and pitch. Since roll and pitch are practically equal to zero, the simplified Jacobian is adequate for this study. The choice of Jacobian, however, does not impose any limitations on the scalability of the navigation algorithm to the full 6-dof problem.

4.4.5 State Integration

Global velocities are integrated to yield the path that was followed by the robot. The integration of linear velocities is straightforward and can be performed using a standard numerical scheme. Integrating the angular velocity, however, requires some special attention. This problem originates from the topology of the set of rotations. The set of rotations forms an algebraic group, namely the special orthogonal group. The special orthogonal group (SO(3)) is closed under a composition operation (e.g. for the case of rotation matrices the group operation is the matrix multiplication). For this reason, numerical schemes involving addition cannot be applied directly; addition is not an admissible operation within the group. To remedy that, the integration problem is transformed to another space. The indicated space is the tangent space of the SO(3), namely the Lie Algebra $\mathfrak{so}(3)$. The Lie algebra is a vector space; hence, any standard integration scheme can be applied. The outcome of the integration is transformed back to the SO(3) using the exponential map. Integration within $\mathfrak{so}(3)$ respects the topology of SO(3). Therefore, the result of the integration is guaranteed to be a valid rotation. A detailed treatise on the subject can be found in [175].

4.5 Experiments

The focal point of this section is to evaluate the performance of STKF in the presence of sensor failures. To this end, all three sensor failures, as mentioned previously, have been simulated: 1) the sensor ceases to send any measurements (i.e., the sensor is offline), 2) the sensor constantly outputs the same value, and 3) the sensor measurement is corrupted by random high-variance noise (outliers) with probability p. The AUV can sustain mission execution when the proposed algorithm estimates the instantaneous velocity of the vehicle, as if all sensors were functioning properly. The evaluation of the algorithm merely focuses on the gyro and the DVL. During the experiments, the magnetic compass was solely used to get the initial heading of the vehicle. Pressure sensor failures were not considered during the experiments; mainly due to the limited motion of the vehicle along this dimension. Nevertheless, the STKF can be used to account for failures of the pressure sensor as well.

4.5.1 Wave tank experiments

Similarly to the training procedure (see Section 3.3.2), the data for the experimental evaluation were gathered by manually driving Nessie around a wave tank in Heriot-Watt. The information from the navigation sensors was recorded into a ROS bag. Two types of trajectories have been recorded: an eight-shaped trajectory and a circular trajectory. In what follows, the output of the proposed filter is compared against the ground truth (the vehicle's state computed with a standard KF and uncorrupted measurements from the sensors). Moreover, the corrupted measurements from the simulated faulty sensor are used within the standard Kalman Filter; the output of the latter is also provided for comparison.

Specifically, Figure 4.12 shows the instantaneous local velocity of the vehicle along the surge and Figure 4.13 along the sway dimensions respectively. At a certain point in time, the DVL stopped sending measurements to the navigation system. In this case, the dynamic model takes over the computation of the vehicle's velocity. The navigation algorithm is able to approximate accurately the corresponding velocity profiles. Figures 4.14, 4.15 depict the same information for the vehicle's local

velocity when the measurement is corrupted with outliers. The outliers were generated as described previously with probability $p = 0.4$. Likewise, in Figures 4.16, 4.17 the DVL is stuck at a specific value. In this experiment the indicated value is equal to 1 for both the surge and the sway dimensions. Once again the algorithm estimates correctly the instantaneous local velocities of the vehicle. Similar analysis was applied for simulated gyro failures. Figures 4.19, 4.18 depict the angular velocity as estimated by the standard Kalman Filter and the proposed navigation algorithm for the stuck sensor case, as well as when the gyro is affected by outliers. The gyro was stuck at a constant value of 1 rad/sec. Obviously, the STKF outperforms the standard Kalman Filter in the estimation of the vehicle's velocity in all the failure scenarios discussed above.

In the last experiment, two sensors failed simultaneously; namely, the DVL measurement was corrupted with outliers, whereas the gyro is stuck to a constant value. The model has been used appropriately to compensate for the broken sensors. As shown in Figures 4.20, 4.21, 4.22 the navigation system was able to compute the respective velocities for all three dimensions (surge, sway, yaw rate).

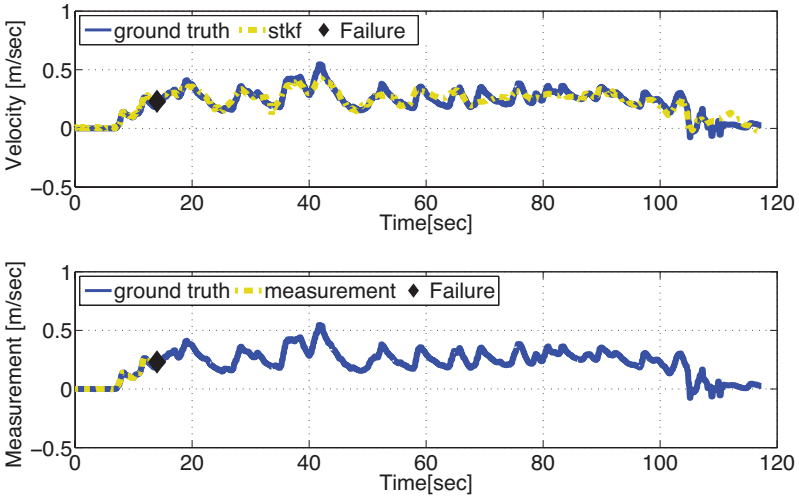


Figure 4.12: The top graph compares the estimated surge velocity with the respective DVL measurement. The bottom graph illustrates the stream of measurements that was available navigation algorithm. The RMSE of the algorithm's estimation for this experiment is equal to 0.041 m/sec

Figure 4.23 summarises STKF's performance for all the experiments that were undertaken in the wave tank.

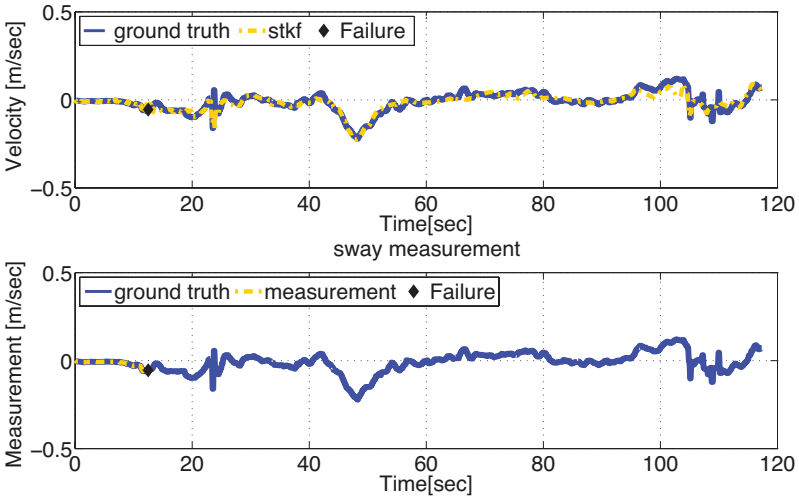


Figure 4.13: The top graph compares the estimated sway velocity with the respective DVL measurement. The bottom graph illustrates the stream of measurements that was available navigation algorithm. The RMSE of the algorithm's estimation for this experiment is equal to 0.024 m/sec

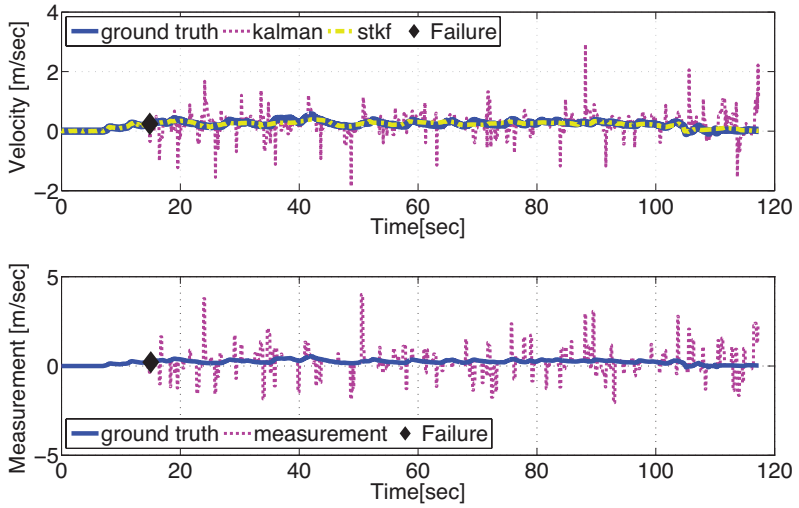


Figure 4.14: The top graph compares the estimated surge velocity with the respective DVL measurement. Moreover, the output of the standard Kalman Filter is also given for comparison. The bottom graph illustrates the stream of measurements that was available both to our navigation algorithm and the Kalman Filter. The RMSE for this experiment is equal to 0.048 m/sec

4.5.2 CMRE experiments

Further to the wave tank experiments, the algorithm was also tested on data gathered in the arena of the Student Autonomous Underwater Competition - Europe (SAUC-E). The competition took place in the Centre of Marine Research (CMRE) in La Spezia, Italy. The data from the competition arena constitute a much harder test for the navigation algorithm; for the competition arena is a less controlled en-

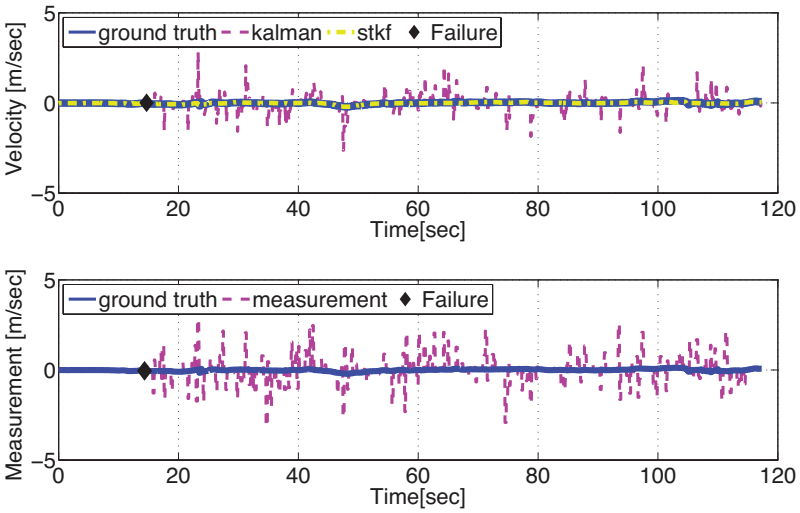


Figure 4.15: The top graph compares the estimated sway velocity with the respective DVL measurement. Moreover, the output of the standard Kalman Filter is also given for comparison. The bottom graph illustrates the stream of measurements that was available both to our navigation algorithm and the Kalman Filter. The RMSE for this experiment is equal to 0.0248 m/sec

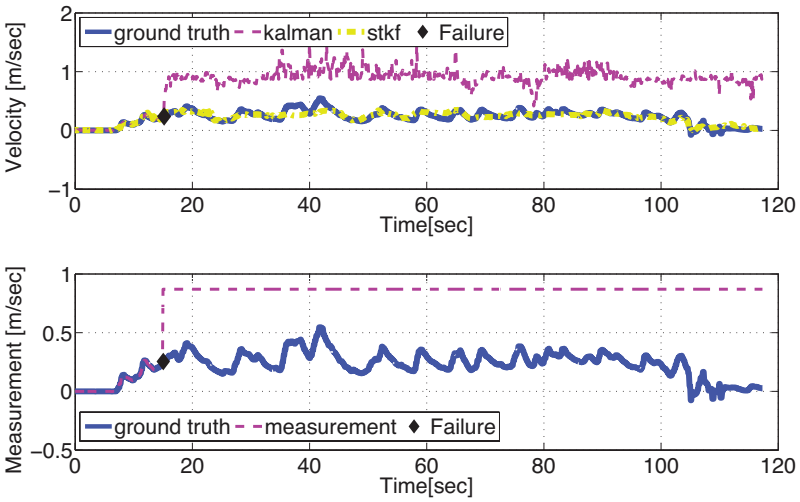


Figure 4.16: The top graph compares the estimated surge velocity with the respective DVL measurement. Moreover, the output of the standard Kalman Filter is also given for comparison. The bottom graph illustrates the stream of measurements that was available both to our navigation algorithm and the Kalman Filter. The RMSE for this experiment is equal to 0.055 m/sec

vironment than the wave tank. Moreover, the dynamic model of the vehicle has not been retrained for this test. The reason for that was that the recorded data from the competition were limited; hence, they were used merely for testing and not for calibrating the model. Furthermore, the sensitivity of the navigation algorithm to the model accuracy can be tested in this manner.

Two different experiments have been conducted on the CMRE data. Firstly,

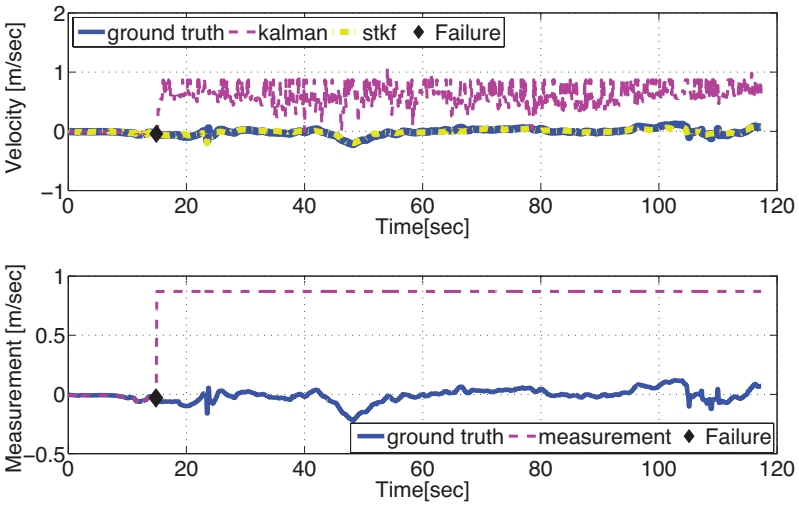


Figure 4.17: The top graph compares the estimated sway velocity with the respective DVL measurement. Moreover, the output of the standard Kalman Filter is also given for comparison. The bottom graph illustrates the stream of measurements that was available both to our navigation algorithm and the Kalman Filter. The RMSE for this experiment is equal to 0.025 m/sec

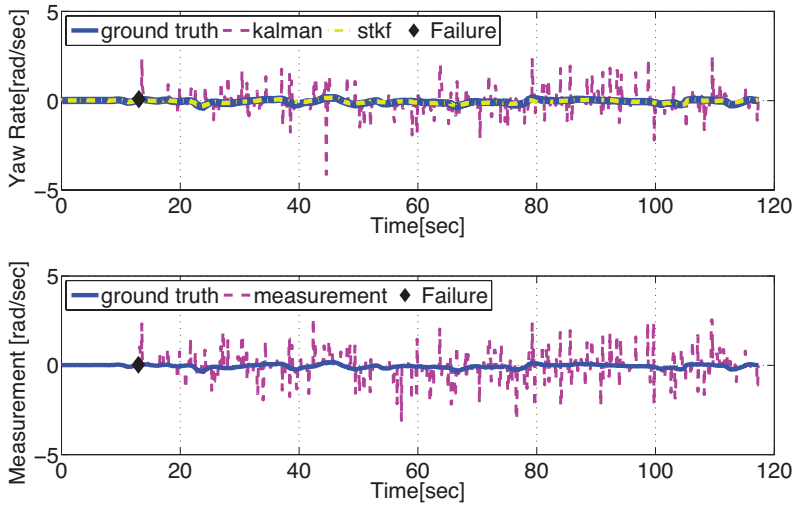


Figure 4.18: In this experiment the gyro measurement was corrupted with outliers. The outlier generation probability was equal to 0.4. The fault occurred at a certain time instant denoted by a black diamond. The top graph compares the estimated yaw rate with the actual gyro measurement. Moreover, the output of the standard Kalman Filter is also given for comparison. At the bottom the actual measurement and the measurement after simulating the failure are provided. This experiment took place in the wave tank in Heriot-Watt. The RMSE for this experiment is equal to 0.19 rad/sec

the DVL was stuck at a constant zero value both along the surge and the sway dimension. As shown in Figure 4.24, the algorithm estimates the velocities less accurately, albeit still quite closely. The accuracy loss, compared to the equivalent wave tank experiments, arises from the uncalibrated dynamic model. Nevertheless,

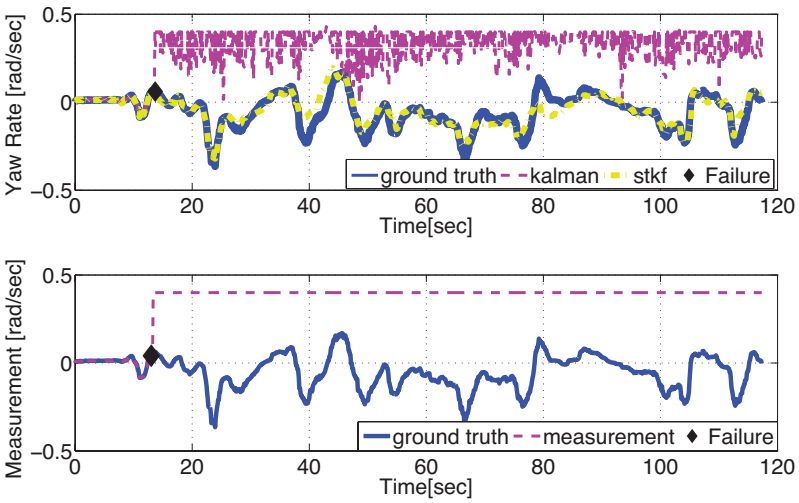


Figure 4.19: In this experiment the gyro measurement was stuck at a constant value of 0.4 rad/sec. The fault occurred at a certain time instant denoted by a black diamond. The top graph compares the estimated yaw rate with the actual gyro measurement. Moreover, the output of the standard Kalman Filter is also given for comparison. At the bottom the actual measurement and the measurement after simulating the failure are provided. This experiment took place in the wave tank in Heriot-Watt. The RMSE for this experiment is equal to 0.178 rad/sec

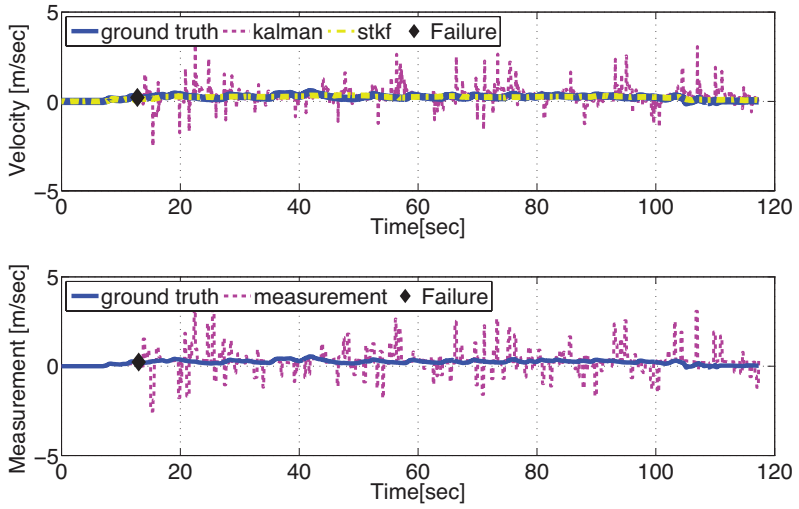


Figure 4.20: The top graph compares the estimated surge velocity with the respective DVL measurement. Moreover, the output of the standard Kalman Filter is also given for comparison. The bottom graph illustrates the stream of measurements that was available both to our navigation algorithm and the Kalman Filter. The RMSE for this experiment is equal to 0.073 m/sec

the navigation algorithm still outperforms the standard Kalman Filter. In the second experiment, the gyro measurement was corrupted with outliers (Figure 4.26). The estimation of the vehicle's angular velocity is not influenced by the presence of outliers.

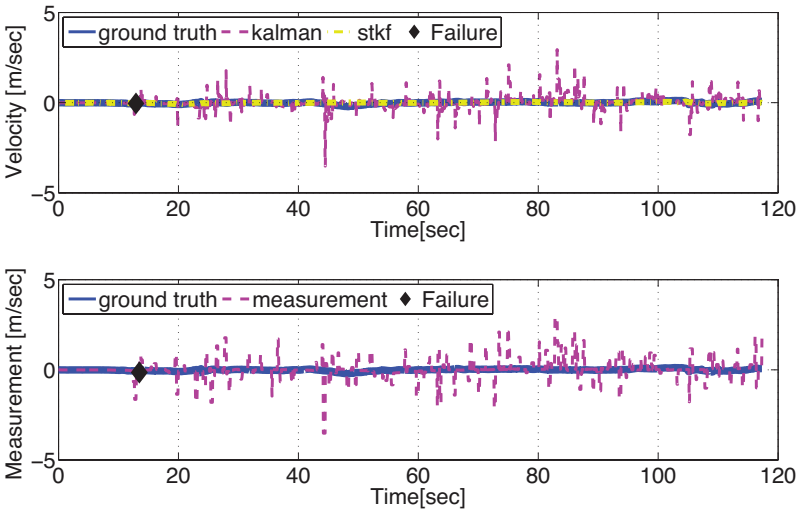


Figure 4.21: The top graph compares the estimated surge velocity with the respective DVL measurement. Moreover, the output of the standard Kalman Filter is also given for comparison. The bottom graph illustrates the stream of measurements that was available both to our navigation algorithm and the Kalman Filter. The RMSE for this experiment is equal to 0.05 m/sec

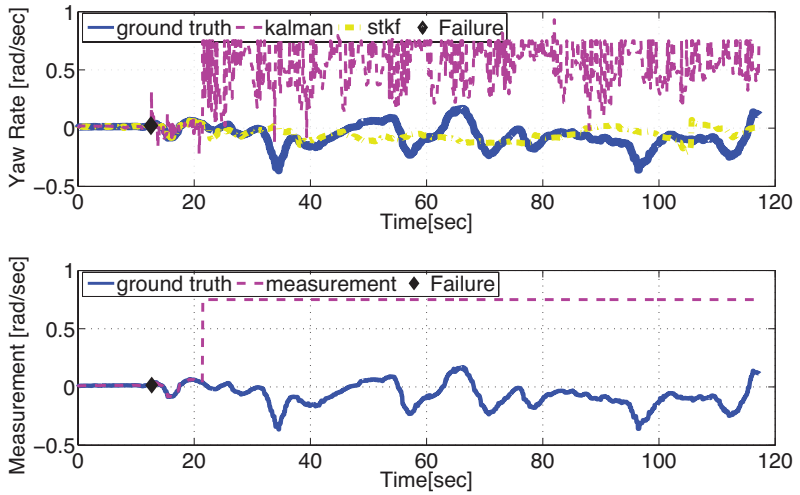


Figure 4.22: The top graph compares the estimated yaw rate with the actual gyro measurement. Moreover, the output of the standard Kalman Filter is also given for comparison. The bottom graph illustrates the stream of measurements that was available both to our navigation algorithm and the Kalman Filter. The RMSE for this experiment is equal to 0.165 rad/sec

4.5.3 Trajectory computation

Following the velocity estimation, the navigation algorithm converts the local velocities to the global coordinate frame. Furthermore, the velocities are integrated appropriately (see Section 4.4.5) to yield the path of the robot. During integration, small errors in the velocity estimation accumulate and result to paths that differ from the ground truth. Given the absence of absolute position measurements,

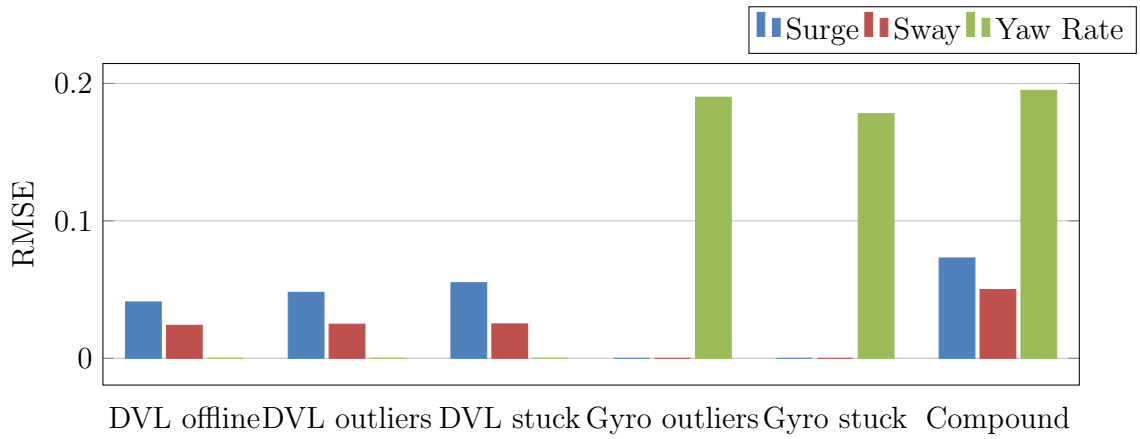


Figure 4.23: Wave tank experiment summary. The surge and sway RMSE is in [m/sec], whereas for the yaw rate in [rad/sec]. The compound fault in the last experiment was simulated by introducing outliers to the DVL, while in parallel the gyro's output got stuck at a constant value.

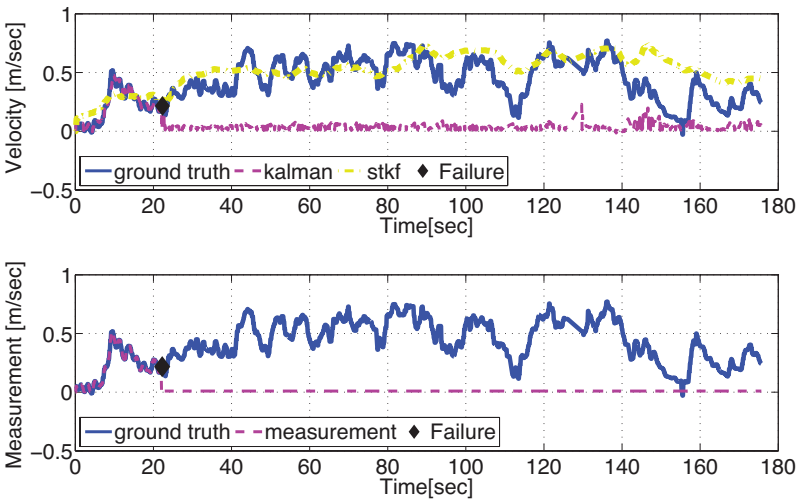


Figure 4.24: The top graph compares the estimated surge velocity with the respective DVL measurement. Moreover, the output of the standard Kalman Filter is also given for comparison. The bottom graph illustrates the stream of measurements that was available both to our navigation algorithm and the Kalman Filter. The RMSE for this experiment is equal to 0.17 m/sec

which could be used to correct the drift, this artifact cannot be avoided. In the remaining of this section, four different experiments and the respective trajectories are discussed.

In Figure 4.27, the DVL stops to send measurements to the navigation algorithm. However, the algorithm compensates by using the dynamic model of the vehicle. The navigation output approximates the actual robot path accurately, as computed when the DVL operated correctly (ground truth). Figure 4.28 illustrates the same trajectory comparison when the DVL measurements are corrupted with outliers.

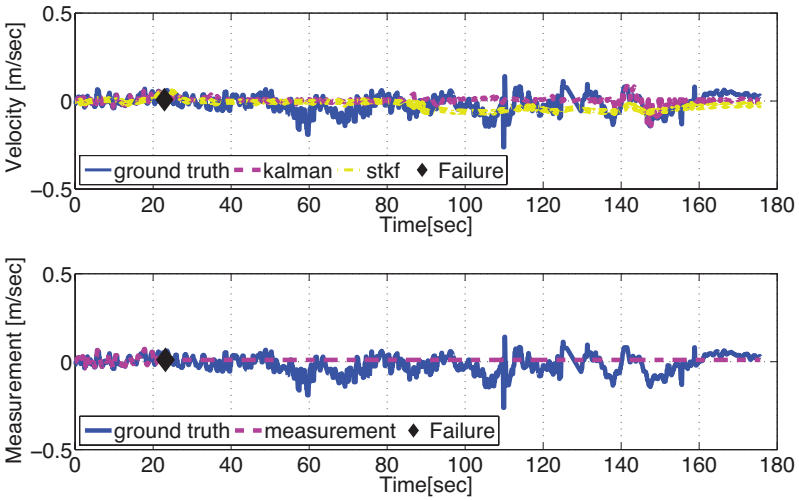


Figure 4.25: The top graph compares the estimated surge velocity with the respective DVL measurement. Moreover, the output of the standard Kalman Filter is also given for comparison. The bottom graph illustrates the stream of measurements that was available both to our navigation algorithm and the Kalman Filter. The RMSE for this experiment is equal to 0.05 m/sec

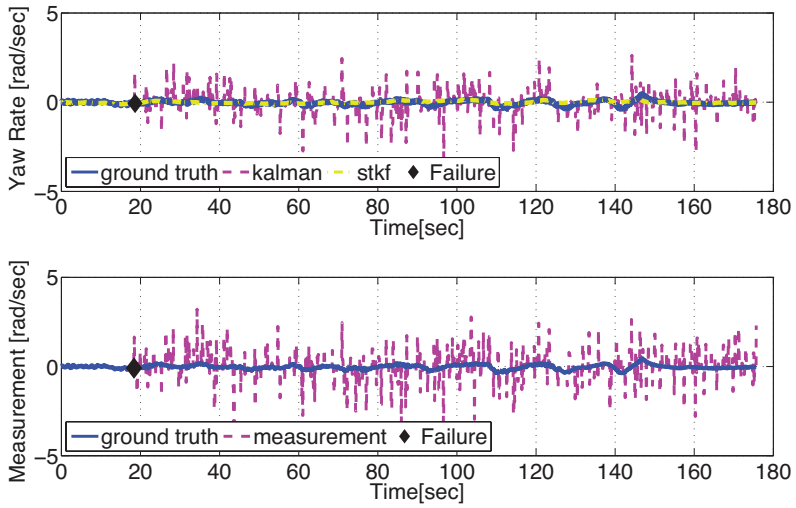


Figure 4.26: In this experiment the gyro measurement was corrupted with outliers. The outlier generation probability was equal to 0.2. The fault occurred at a certain time instant denoted by a black diamond. The top graph compares the estimated yaw rate with the actual gyro measurement. Moreover, the output of the standard Kalman Filter is also given for comparison. At the bottom the actual measurement and the measurement after simulating the failure are provided. This experiment took place in the SAUC-E competition arena at CMRE. The RMSE for this experiment is equal to 0.16 rad/sec

Also in this case the trajectory is computed accurately. The third experiment, shown in Figure 4.29, compares the estimated trajectory with the ground truth when the DVL is stuck to a constant value. In this experiment, a different trajectory has been used. The indicated trajectory was recorded in the same wave tank, approximately

one year after the previous experiment. The dynamic model was re-calibrated to account for any possible changes in the robot's dynamics. This experiment involved a more complex and longer trajectory. Therefore, the accumulated drift is more noticeable in the final result. Finally, Figure 4.30 depicts the estimated trajectory in the case of outliers, using the data gathered at CMRE's competition arena. The model was not re-calibrated, hence the stronger presence of drift in the final path estimation.

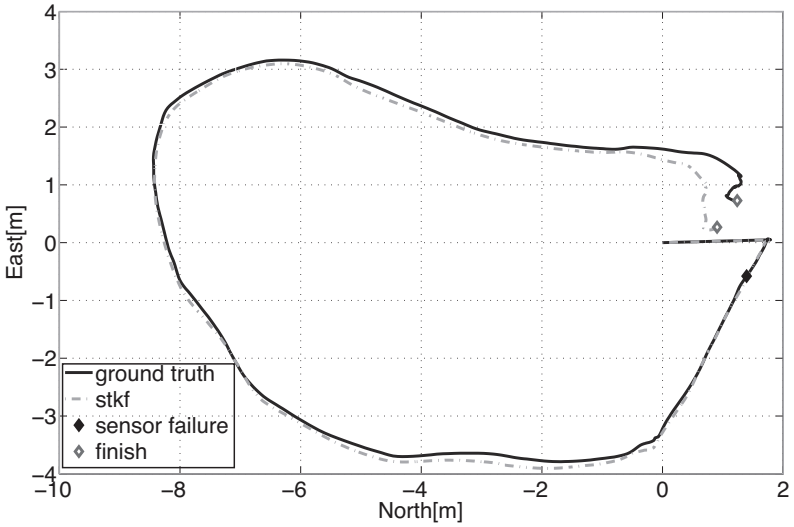


Figure 4.27: Comparison of the robot's estimated path, as computed by integrating the output of the proposed navigation algorithm, with the path computed by integrating appropriately the actual DVL measurements (ground truth). In this experiment the DVL went suddenly offline (black diamond). The experiment took place in the wave tank in Heriot-Watt

4.6 Sensor Diagnostics

Further to the estimation of the velocity and position of the robot, a basic reasoning system has been developed to aid fault diagnostics. The diagnostic system is triggered whenever an outlier is rejected by the filter. For the detection of an outlier, a statistical test has been used. In detail, the diagonal elements of the resulting measurement covariance $\mathbf{R}_k$ are compared to the nominal values $\mathbf{R}_{init}$, as found in

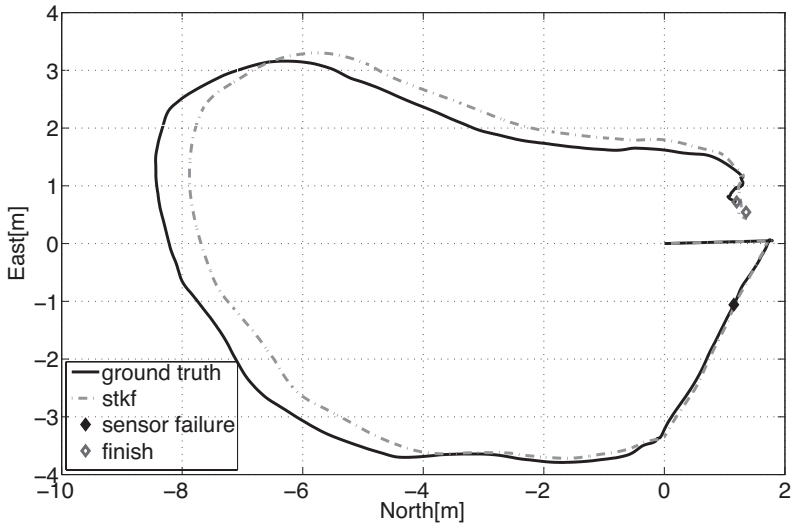


Figure 4.28: Comparison of the robot's estimated path, as computed by integrating the output of the proposed navigation algorithm, with the path computed by integrating appropriately the actual DVL measurements (ground truth). In this experiment the DVL measurement was corrupted with outliers with probability $p = 0.4$. The experiment took place in the wave tank in Heriot-Watt

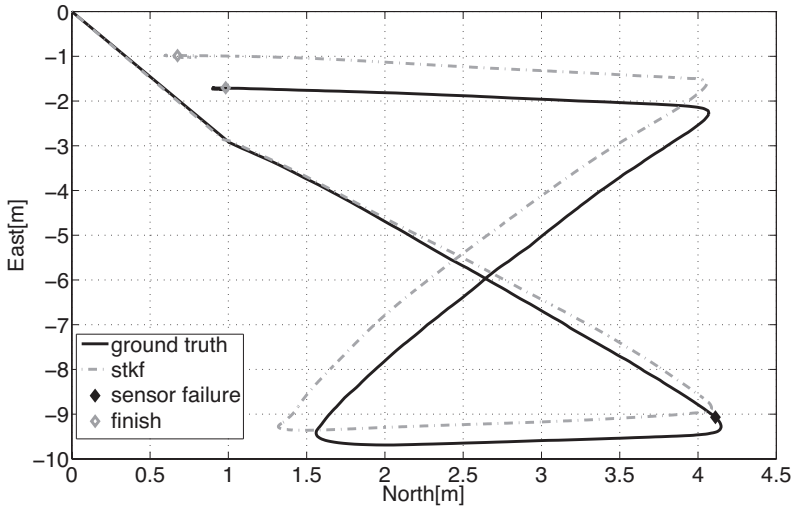


Figure 4.29: Comparison of the robot's estimated path, as computed by integrating the output of the proposed navigation algorithm, with the path computed by integrating appropriately the actual DVL measurements (ground truth). In this experiment the DVL measurement was stuck to a constant value of zero. The experiment took place in the wave tank in Heriot-Watt

the sensor specifications. Outlier detection is formulated as an F-test:

$$H_0 : \sigma^2_{spec} = \sigma^2_{est} \tag{4.39}$$

$$H_1 : \sigma^2_{spec} < \sigma^2_{est}$$

$$F = \frac{\sigma^2_{spec}}{\sigma^2_{est}}$$

$$\alpha : 0.99$$

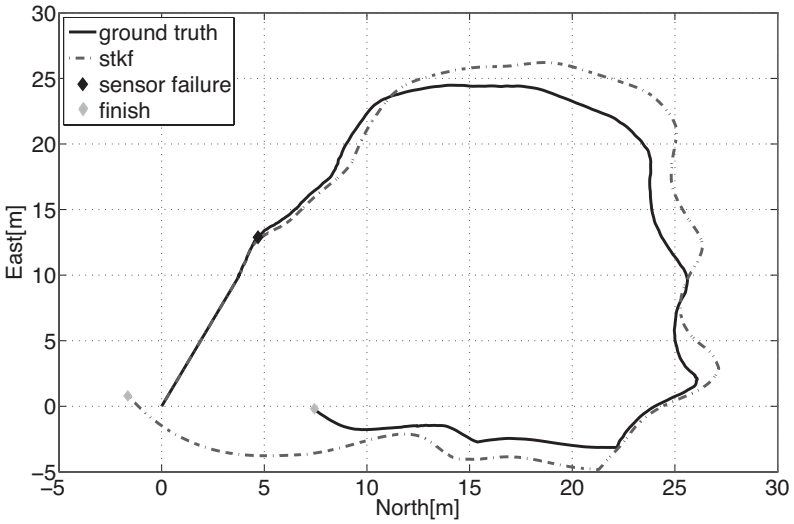


Figure 4.30: Comparison of the robot's estimated path, as computed by integrating the output of the proposed navigation algorithm, with the path computed by integrating appropriately the actual DVL measurements (ground truth). In this experiment the DVL measurement was corrupted with outliers with probability $p = 0.2$. The experiment took place in the SAUCE-E competition arena in CMRE.

The null hypothesis of the test, namely H_0, is that the variances of the original (σ_{spec}) and the computed measurement models (σ_{est}) are not statistically different. A confidence level of $\alpha = 0.99$ has been chosen. The null hypothesis is rejected whenever $F < F_{1-\alpha}$, where $F_{1-\alpha}$ is the 1%-percentile of the F distribution. When H_0 is rejected, an outlier has been detected.

Following the outlier detection, the diagnostic system performs inference to distinguish between the failure modes. Specifically, the system observes the outlier detection events and estimates the probability of each failure mode. Diagnostic inference does not include the case when the sensor has gone offline; it is fairly easy to distinguish it from the other failure modes using the timestamps of the measurements. Consequently, the considered health states are *healthy*, *outlier-corrupted* and *sensor-stuck*. A Hidden Markov Model (HMM, [116]) is used to model the probability distribution of the state, conditioned on the observation sequence. In this example, the observation sequence consists of outlier detection events (erroneous measurements). An HMM is defined by the initial probabilities of the state, the transition probabilities and the emission probabilities. The estimation of the HMM properties is described in Section 4.6.1.

4.6.1 HMM training

A Hidden Markov Model is parametrised by the initial probability distribution of the state π, the transition model $\mathcal{T}$, and the observation model $\mathcal{O}$ (emission model). The transition model $\mathcal{T}$ is an $n \times n$ matrix, where n is the cardinality of the HMM's state. The ij-th entry of the transition model holds the probability of transition from state i to state j. The emission model is an $n \times m$ matrix, where m is the cardinality of the observation set. The ij-th entry of the emission model is the probability that the j-th value of the observation variable will be observed when in state i.

The diagnostic HMM was trained on synthetic data using the Baum-Welch algorithm. This particular training method requires initial guesses for the transition and the observation models, as well as a training observation sequence. The observation sequence was generated by the following rules:

- A healthy system generates outliers with probability $p \to 0$
- A stuck sensor generates outliers with probability $p \to 1$
- A system that suffers from random outliers will generate corrupted measurements with probability $p \sim \mathcal{U}(0, 1)$.

The system tends to preserve the state, i.e., the non-diagonal entries of the transition model are very small, albeit non-zero. Consequently, the diagonal entries of the transition model are close to one. The transition and the emission matrices as computed after training the HMM are:

$$\mathcal{T} = \begin{bmatrix} 0.9986 & 0.0013 & 0.0001 \\ 0.002 & 0.998 & 0.000 \\ 0.002 & 0.000 & 0.998 \end{bmatrix}, \mathcal{O} = \begin{bmatrix} 0.99 & 0.01 \\ 0.672 & 0.327 \\ 0.01 & 0.99 \end{bmatrix} \tag{4.40}$$

The resulting HMM was evaluated on a separate set of observation sequences. The model is able to estimate the failure mode correctly in the vast majority of the test sequences. Table 4.1 depicts the confusion matrix of the HMM-based diagnostic system:

Table 4.1: Confusion Matrix of the Diagnostics System

		Predicted Class		
		Healthy	*Outlier*	*Stuck*
Actual Class	*Healthy*	98	2	0
	Outlier	5	73	22
	Stuck	0	0	100

As described above, the algorithm performs a statistical test on the diagonal elements of the sampled measurement covariance $\mathbf{R}_k$ with respect to the corresponding values of $\mathbf{R}_{init}$. Rejection of the null hypothesis H_0 indicates the detection of an outlier. The detection is logged in the event channel, accompanied by a timestamp and the name of the sensor that triggered the event. The log is then given as input to the HMM (Section 4.6.1) to infer the failure mode. The event channel is used as the observation sequence of the HMM. Table 4.2 summarizes the probability distributions of the failure modes for the four experiments discussed in Section 4.5.3.

Table 4.2: Failure Mode Probabilities

Experiment	*healthy*	*outliers*	*stuck*
Loop # 1	0.987	0.013	0
Loop # 2	0.143	0.651	0.206
8-shaped	0	0.01	0.99
cmre	0	0.317	0.683

As mentioned earlier, isolating the offline failure mode is trivial. Furthermore, the experiments have shown that the HMM isolates the *stuck-sensor* failure mode successfully. Confusion arises in the case of random outliers. The probability at which the outliers are generated influences the performance of the diagnostic system. Specifically, the diagnostic system may misclassify the sensor as healthy, when the outlier probability is too low, or stuck when the outlier probability is too high. For a certain range of p (e.g. $0.15 - 0.70$) the diagnostics are correct. The described behavior is quite intuitive; a human operator would possibly encounter a similar problem. Nevertheless, a significant improvement is expected by adding more complex features.

4.7 Remarks

This chapter presented two algorithms for outlier rejection for non-linear non-stationary processes. The first filter is based on [1]. The filter has been derived from scratch; i.e., by formulating the problem in terms of a Bayesian Network, and solving for the hidden variables. The yielded algorithm is similar to the EKF; in the sense that the non-linear dynamics are linearised using the Jacobian. However, the state covariance is updated slightly differently than standard EKF. Moreover, a set of recursive equations has been provided, to tune the required parameters of the filter.

The second filter, namely STKF, is a direct combination of EKF and ORKF [152]. The resulted is more robust than the first presented algorithm. This is mostly because, no structure is imposed on the sampled measurement covariance; any positive semi-definite matrix is possible. All dimensions are handled separately, based on the quality of the incoming sensory information.

A large contribution to both algorithms' performance is attributed to LWPR. The powerful dynamic modelling, as described in Chapter 3, is the keynote for the algorithms' success. It has been shown on real-world data that outlier rejection, as shown by [152], and [1] for stationary processes, can be extended to highly complex systems, like an autonomous underwater vehicle.

Finally, a simple sensor diagnostic approach enables to discriminate between sensor failures. Specifically, the diagnostic system detects whether the sensor is stuck or if the measurement sequence is corrupted by outliers. Naturally, there is some ambiguity in the switching boundaries between failure modes. To remedy that, the system could leverage higher level information from other monitoring subsystems or to provide more representative training data to the system.

Chapter 5

Robustness to Changes in the Process Dynamics

5.1 Introduction

Over the course of time, autonomous systems change dynamic behaviour. Changes occur either gradually or abruptly. Asset ageing (i.e. wear) causes gradual changes; whereas, abrupt changes occur after hardware failure (e.g., after a thruster failure in the case of AUVs). In this chapter, an algorithm is presented that can detect shifts in the dynamics of an autonomous system.

Similarly to sensor failures, a Bayesian filter is utilised to detect deviations from normal operation (i.e., faults). Next, deviations are included in a new dynamic model; namely, a model that captures the system dynamics after failure. In this way, the algorithm considers the fault in future state estimation. The algorithm can be applied in the presence of multiple unknown faults; albeit, the rest of this chapter considers the case of a single fault.

The proposed algorithm uses two models: a *nominal* and an *adaptive* model. The nominal model represents normal operation; whereas, the adaptive model captures the system's dynamics after a fault has occurred. Initially, the adaptive model is identical to the nominal. During operation, however, the adaptive model incorporates incoming sensory information continuously. When the system drifts from normal operation, predictions by the two models get different. It is important to

mention that the adaptive model considers only seminal changes in the dynamics. The Bayesian formulation filters out noise and outliers in the measurements sequence. What remains is used as input to the adaptive model.

Given a new measurement, the algorithm infers which of the two models explains the observations better. This is realised using a discrete-valued *switch* variable $s \in \mathbb{R}^k$, where k is the number of different dynamic models. When the k-th model is active, the respective entry in the switch variable becomes one; all The algorithm computes the joint probability of the vehicle's state (e.g., velocity and acceleration) and the switch variable y using a variational approximation.

During normal operation, both the nominal and the adaptive model will output similar state estimations; hence, in this case, both models are equiprobable ($p(s_1 = 1) = p(s_2 = 1) = 0.5$). When something modifies the vehicle's dynamics substantially, however, the adaptive model will yield the most plausible explanation; hence in this case $p(s_2 = 1)$ gets larger. In such cases, the adaptive model will reflect the actual dynamics of the system after the fault. Next, the expected value of the vehicle's state is computed as the weighted sum of the output of the two models; using as weights the probability distribution of s. Assuming Gaussian distributions for the individual model predictions, the state expectation is, in fact, a *Mixture of Gaussians*.

This chapter provides a mathematically rigorous framework for fault detection and online model adaptation. The Bayesian representation naturally precludes the problem of uncertainty. No explicit parameters (e.g. thresholds) need to be defined to indicate faulty behaviour or to perform outlier detection. Expert knowledge is provided through *prior distributions*. The latter are intuitive to choose and commonly shared across all model dimensions. Moreover, the prior parameters can be learned from data using a *Maximum Likelihood* approach; albeit this has not been undertaken in this brief.

A key contribution of the suggested algorithm is that the *adaptive* model captures arbitrary deviations from the dynamics without the need to predefine the fault dynamics. At each time, an up-to-date model that can be used for control and plan-

ning is provided. Moreover, the presented algorithm encompasses previous model detection methods; for it can detect faults with unknown dynamics. Nevertheless, any model of a previously known fault can be simply added to the model ensemble (i.e., the set of models that define the mixture of Gaussians). In a way, the algorithm presented here extends standard model-based fault detection from a set of predefined faults to the full space of possible faults.

5.2 Fault Detection and Dynamic Adaptation Algorithm

The algorithm is an extension of the standard Kalman filter to include multiple possible models. Not all models are known a priori; conversely, they are the outcome of continuous model adaptation. In the graphical representation, $\mathbf{X}_t$ is the generalised state; i.e., the state as computed by weighing together all model predictions at time t, $\mathbf{Y}_t$ holds the sensor data at the same time instant. $\mathbf{\Lambda}_t$ is the prior for the sensor precision (i.e., the inverse of sensor covariance; sensor noise model of the Kalman equivalent) and, lastly, $\mathbf{S}_t$ is the switch variable. To simplify the Bayesian Network, no prior for the prediction covariance has been utilised. Since merely the relative magnitude of the model and measurement covariance is important, random variable $\mathbf{\Lambda}_t$ is sufficient. Whenever index t is omitted, the variable holds the trajectory across time of the corresponding time indexed variable.

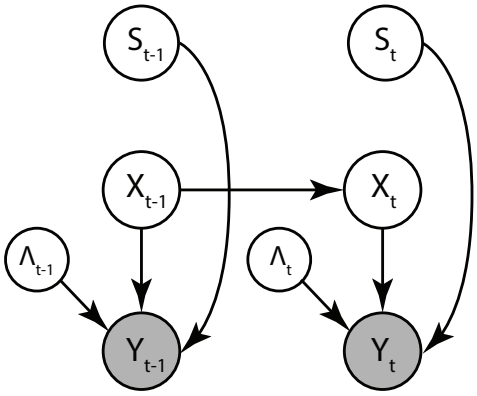


Figure 5.1: Representation of the fault detection algorithm as two-slice Dynamic Bayesian Network.

Using the *Chain Rule for Bayesian Networks* [169], the joint probability depicted

in Figure 5.1 factorises as follows:

$$P(\mathbf{Y}, \mathbf{X}, \mathbf{\Lambda}, \mathbf{s}) = P(\mathbf{Y}|\mathbf{X}, \mathbf{\Lambda}, \mathbf{s})P(\mathbf{s}|\pi)P(\mathbf{\Lambda})P(\mathbf{X}) \tag{5.1}$$

$$P(\mathbf{Y}|\mathbf{X}, \mathbf{\Lambda}, \mathbf{s}) = \prod_{n=1}^{N}\prod_{i=1}^{M}\mathcal{N}(\mathbf{y}_n|\mathbf{C}\mathbf{x}_n^{(i)}, \mathbf{\Lambda}_n^{(i)})^{s_n^{(i)}} \tag{5.2}$$

$$P(\mathbf{s}|\pi) = \prod_{n=1}^{N}\prod_{i=1}^{M}\pi_i^{s_n^{(i)}} \tag{5.3}$$

$$P(\mathbf{\Lambda}) = \prod_{n=1}^{N}\prod_{i=1}^{M}\mathcal{W}(\mathbf{\Lambda}_n^{(i)}|\nu^{(i)}, \mathbf{V}^{(i)}) \tag{5.4}$$

$$P(\mathbf{X}) = \prod_{n=1}^{N}\prod_{i=1}^{M}P(\mathbf{x}_n^{(i)}|\mathbf{x}_{n-1})P(\mathbf{x}_0) \tag{5.5}$$

In Equations (5.1)-(5.5), t indexes variables over time, whereas i runs over the set of candidate models (only two in this case; the nominal and the adaptive model). $\mathbf{C}$ is the matrix representation of a linear measurement model, as in the standard Kalman Filter. Moreover, π is the initial *mixing* coefficients; i.e., the initial activation values of each of the models in the ensemble. $\mathcal{W}(.|\nu, \mathbf{V})$ is the Wishart distribution with ν degrees of freedom and seed matrix $\mathbf{V}$ [173]. Applying the Variational Bayes Approximation the final filtering equations are yielded:

$$\langle s_n^{(i)} \rangle = \frac{\tilde{p}_n^{(i)}}{\sum_{i=1}^{M} \tilde{p}_n^{(i)}} \tag{5.6}$$

$$\tilde{p}_n^{(i)} = exp\{\langle ln|\mathbf{\Lambda}_n^{(i)}|\rangle/2 + ln\pi_i - \frac{1}{2} tr\{\langle \mathbf{\Lambda}_n^{(i)} \rangle \langle \alpha_n^{(i)} \rangle\}\} \tag{5.7}$$

$$\langle \alpha_n^{(i)} \rangle = \mathbf{y}_n \mathbf{y}_n^T - \langle \mathbf{x}_n^{(i)} \rangle \mathbf{y}_n^T - \mathbf{y}_n \langle \mathbf{x}_n^{(i),T} \rangle + \langle \mathbf{x}_n^{(i)} \mathbf{x}_n^{(i),T} \rangle \tag{5.8}$$

$$\langle \mathbf{x}_n^{(i)} \rangle = \mathbf{\Lambda}_{x_n}^{(i)} \left[\langle s_n^{(i)} \rangle \langle \mathbf{\Lambda}_n^{(i)} \rangle \mathbf{y}_n + \beta \langle f^{(i)}(\mathbf{x}_{n-1}^{(i)}, \mathbf{u}_{n-1}) \rangle \right] \tag{5.9}$$

$$\mathbf{\Lambda}_{x_n}^{(i)} = \langle s_n^{(i)} \rangle \langle \mathbf{\Lambda}_n^{(i)} \rangle + \beta \mathbf{I} \tag{5.10}$$

$$\langle \mathbf{\Lambda}_n^{(i)} \rangle = \nu_{\Lambda_n}^{(i)} \mathbf{V}_{\Lambda_n}^{(i)^{-1}} \tag{5.11}$$

$$\nu_{\Lambda_n}^{(i)} = \nu^{(i)} + \langle s_n^{(i)} \rangle \tag{5.12}$$

$$\mathbf{V}_{\Lambda_n}^{(i)^{-1}} = \mathbf{V}^{(i)^{-1}} + \langle s_n^{(i)} \rangle \langle \alpha_n^{(i)} \rangle \tag{5.13}$$

where β is a small scalar (e.g., 0.001) that controls the initial relative confidence between model predictions and measurements. Similarly, $\nu^{(i)}$ and $\mathbf{V}^{(i)}$ are the initial Wishart parameters for the measurement model. These parameters capture any expert knowledge about the system The notation $\langle . \rangle$ indicates the expected value of the included random variable. Lastly $f^{(i)}(\mathbf{x}_{n-1}^{(i)}, \mathbf{u}_{n-1})$ is a functional representation of the i-th model; in this thesis f is always approximated using LWPR.

The filtering equations are coupled. To remedy that an iterative update of the parameters is required. It is worth noting, that the filtering equations realise merely the Expectation step of the E-M. This is done to keep the algorithm simple and more computationally efficient. Regardless, even the expectation step is mathematically guaranteed to improve the solution in each iteration [172]

5.2.1 Model Adaptation

As mentioned before, the algorithm utilises a nominal and an adapted model to infer fault detection. This section focuses on adaptation; namely, on the derivation of the adapted model from the nominal. LWPR adapts incrementally by updating the local regression parameters. In light of new data, the algorithm detects which models are activated; this process is identical to the beginning of the prediction computation.

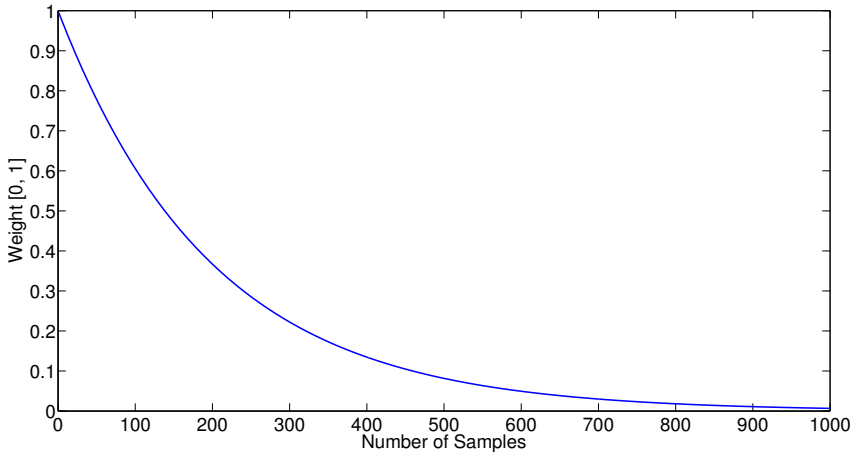


Figure 5.2: Importance weighing of previous training data samples for a forgetting factor $\lambda = 0.995$

The activated models are then updated to incorporate the new information. To allow incremental adaptation of the regression parameters, *sufficient statistics* are used to summarise the information from the previous training data. To control adaptation speed, a *forgetting factor* $\lambda \in [0, 1]$ multiplies the sufficient statistics, regulating the memory behaviour of the system. With $\lambda = 0$, the algorithm disregards the previous training data and updates the parameters solely using the current training sample. A unity forgetting factor locks the regression parameters; yielding a model that ceases to adapt. Choosing the forgetting factor requires attention because extreme values may lead to overfitting or instability. Figure 5.2 shows the relative influence of the previous training samples on the computation of the regression parameters for $\lambda = 0.995$. A detailed description of the regression parameter adaptation can be found in [176].

Fault detection performance depends on robust model adaptation. Robust model adaptation requires that outliers are filtered out, such that detected changes are due to a dynamical shift and not because of noise or sensor defects. To this end, the algorithm adapts the forgetting factor of the underlying LWPR model online, using the current value of the switch variable. Specifically, the probability of $(s_2 = 1)$ is passed into a sigmoid function defined as follows:

$$\lambda = \frac{1}{1 + e^{-\alpha(x-c)}} \tag{5.14}$$

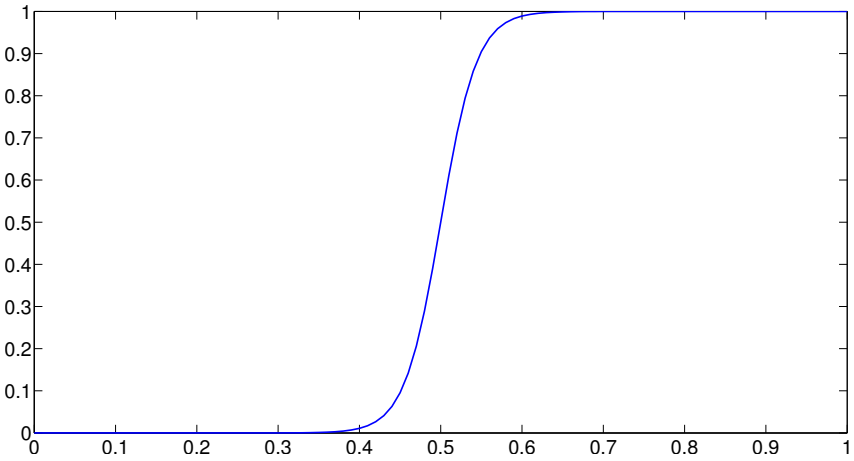


Figure 5.3: Sigmoid function with $\alpha = 10$ and $c = 0.5$.This function regulates the forgetting factor for the model adaptation given the probability $p(s_2 = 1)$

where α is regulates the smoothness of the sigmoid and c the centre. Figure 5.3 depicts the sigmoid used in this application of the algorithm.

5.3 Experimental results

In this section, the presented algorithm is tested on data gathered from the robot. Firstly, the algorithm was applied to data coming from Nessie operating normally. The purpose of this experiment is to check for erroneous fault detections; i.e., cases when the algorithm detects a seminal shift in the autonomous system's dynamics, albeit everything works fine.

Another experiment has been undertaken to test the algorithm's behaviour in the presence of thruster failures. In this experiment, the output of one of Nessie's surge thrusters is limited to a predefined percentage of their nominal thrust capability. The algorithm is supposed to identify the fault through the switch variable s. Specifically, fault detection is performed using a *histogram* the switch variable throughout the experiments. In this way, fault detection takes into consideration all values of s; endowing the algorithm with robustness against random switches due to noise or model imperfections.

Figure 5.4 illustrates the histogram for the activation value of the faulty model (i.e., the second coordinate of the switch variable). Obviously, the mean of the approximated distribution lies closely to 0.5. The resulting histogram is con-

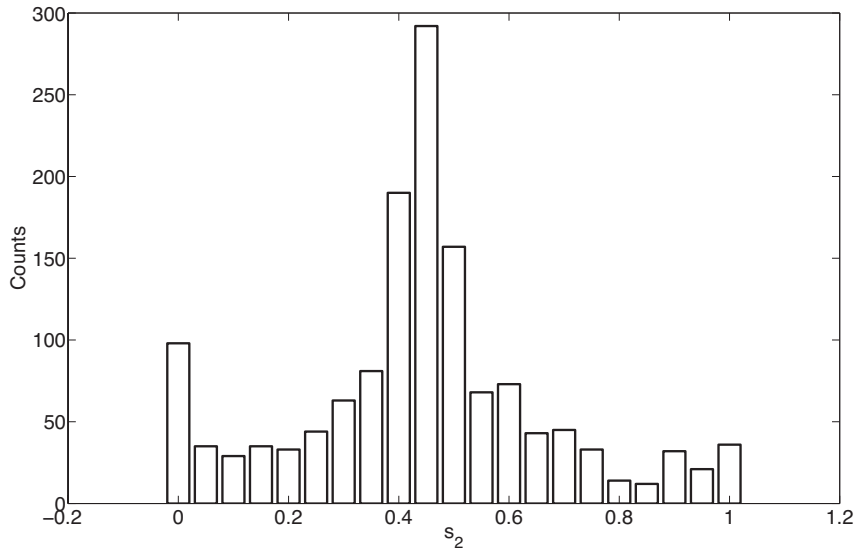


Figure 5.4: Approximation of the probability distribution $p(s_2)$ as a histogram for the activation value of the adaptive model for the healthy case

gruent with the original design; for in the healthy case, observations can be explained equally well by both the healthy and the defective model. Conversely, Figure 5.5 provides the same information in the case of the defective thruster (in this experiment thruster output was limited to 0%; utter thruster failure). In this case, the activation value of the defective model is closer to unity; i.e., the algorithm has detected that the healthy model, no longer explains the observations. Other experiments pertain to limiting the thruster to a non-zero value; i.e., the thruster provides some traction, although not the as much as during healthy operation. Specifically, Figure 5.8 depicts the histogram for s_2 when the thruster output is limited to 68%; similarly, Figure 5.10 for a thruster limit of 51%.

As mentioned previously, the algorithm learns the defective model online. To demonstrate this trait, the internal model predictions (both the healthy and defective) are compared against the measurements in the case of a defective thruster. As seen in Figure 5.6, the adaptive model matches the sensor data closely; i.e., it has learned a new dynamic model to compensate for the thruster failure. The new model can be used to improve either control of planning. Figures 5.7, 5.9 show the respective model predictions for the remaining experiments. Table 5.1 summarises the described experiments in terms of prediction accuracy between the two models. Moreover, the expected value the switch variable is provided for each model:

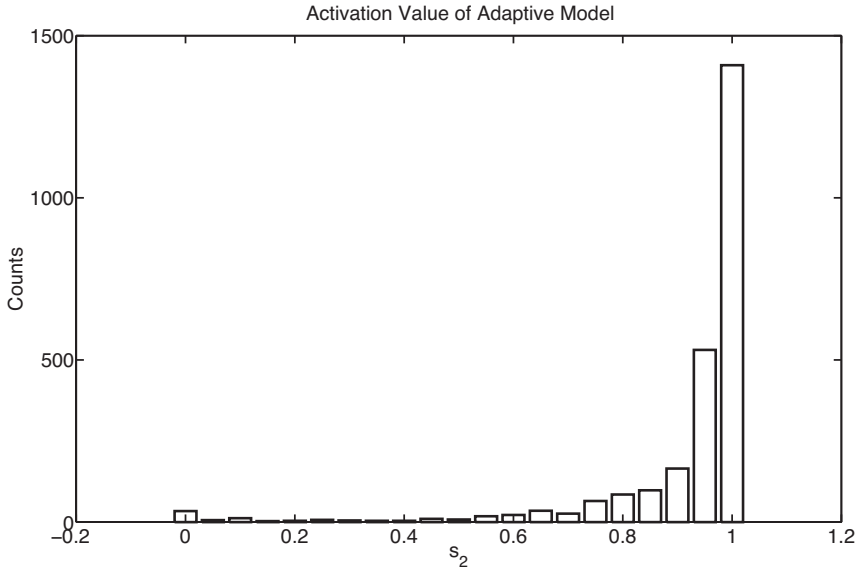


Figure 5.5: Approximation of the probability distribution as a histogram for the activation value of the adaptive model for the defective case (complete thruster failure).

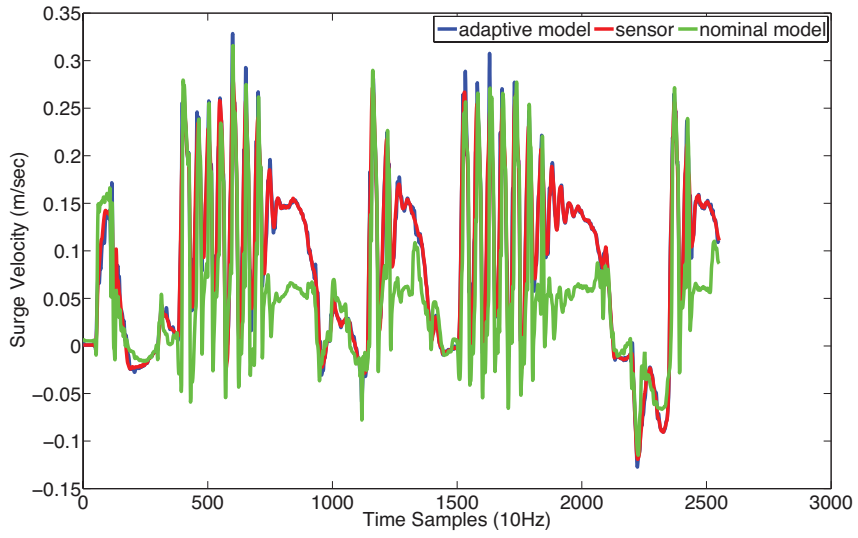


Figure 5.6: Comparison between the nominal and adaptive model prediction. Sensor measurements are utilised to indicate the validity of each model prediction (complete thruster failure).

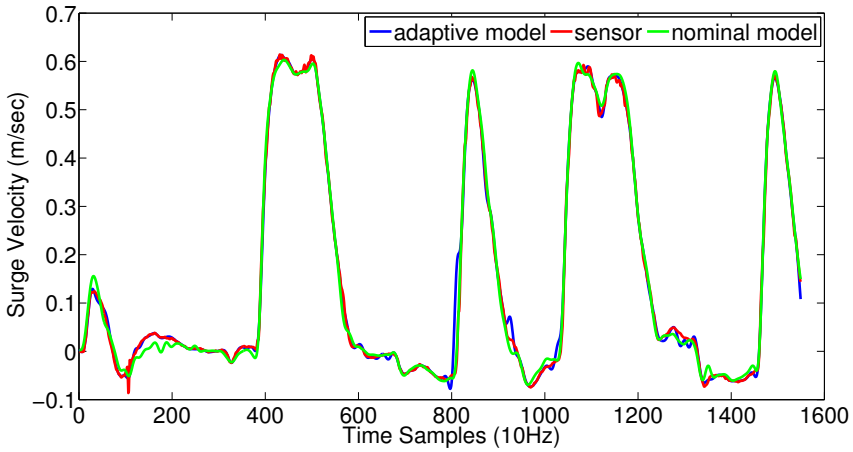


Figure 5.7: Comparison between the nominal and adaptive model prediction. Sensor measurements are utilised to indicate the validity of each model prediction (thruster output limited at 68% of maximum thrust).

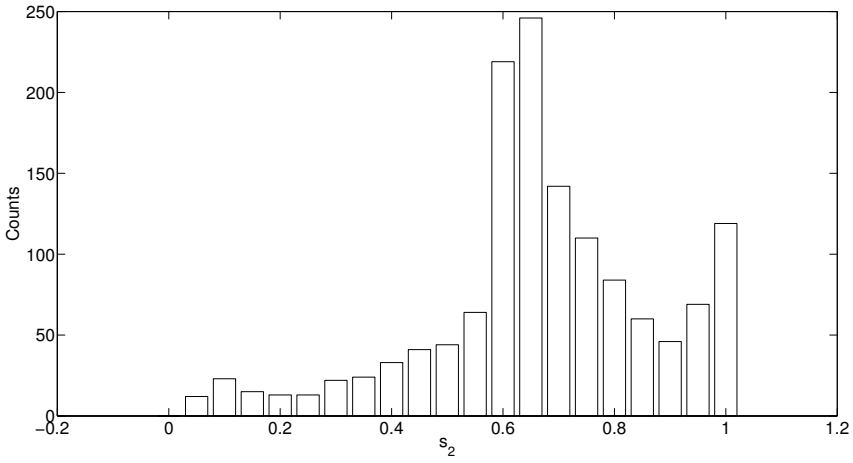


Figure 5.8: Approximation of the probability distribution as a histogram for the activation value of the adaptive model for the defective case (thruster output limited at 68% of maximum thrust).

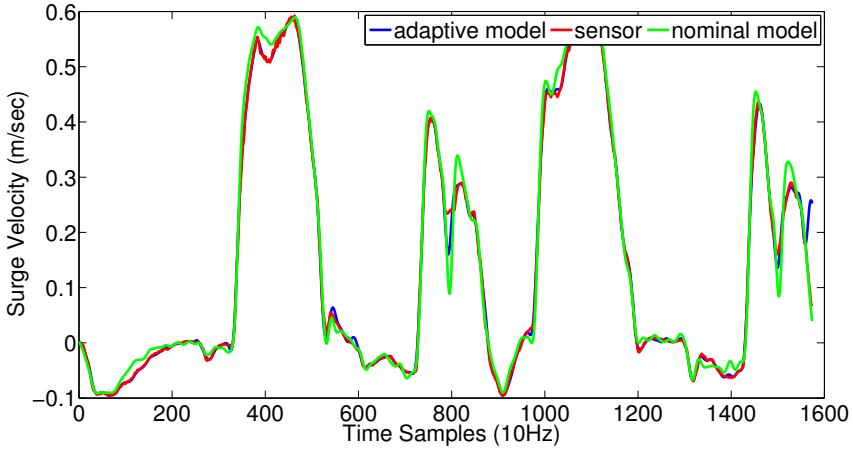


Figure 5.9: Comparison between the nominal and adaptive model prediction. Sensor measurements are utilised to indicate the validity of each model prediction (thruster output limited at 51% of maximum thrust).

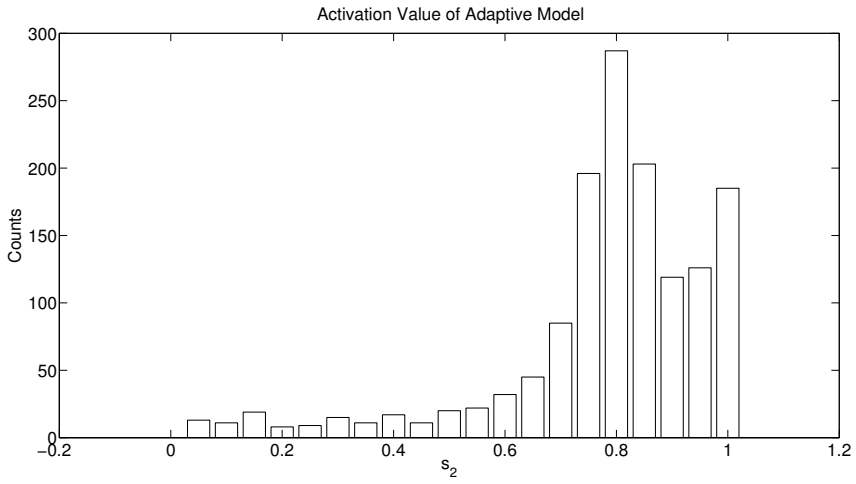


Figure 5.10: Approximation of the probability distribution as a histogram for the activation value of the adaptive model for the defective case (thruster output limited at 51% of maximum thrust)

Table 5.1: Prediction Accuracy and Model Activation

Experiment	Nominal	Adaptive	s_1	s_2
healthy	0.3493	0.4386	0.5592	0.4408
68%	1.4776	0.5561	0.3971	0.6029
51%	1.5008	1.0532	0.2884	0.7116
0%	3.0815	0.9987	0.0819	0.9181

Lastly, the algorithm is tested against sensor failures. The desired behaviour is to detect thruster failures -or any other shift in the autonomous system's dynamics- while preserving the sensor failure robustness. To this end, another experiment has been made by leaving the thruster output unaffected; the sensor measurement has been corrupted with outliers instead. As seen in Figure 5.12 the filtered output is smooth; hence the algorithm's state computation is not influenced by the presence of outliers. In this particular experiment the probability of outlier generation was equal to $p_{outlier} = 0.1$. Also the fault detection remained correct despite the outliers. Figure 5.11 depicts the histogram for s_2; obviously, the mean value of the distribution lies very close to 0.5. This means that the algorithm didn't attribute the corrupted measurement to a subtle changes in the dynamics.

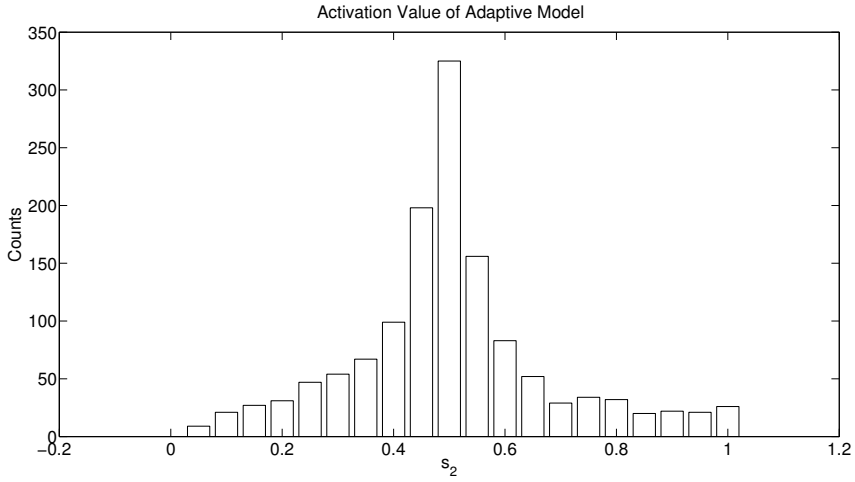


Figure 5.11: Approximation of the probability distribution as a histogram for the activation value of the adaptive model. In this experiment the sensor output was corrupted with randomly generated outliers. The probability of outlier generation is $p_{outlier} = 0.1$.

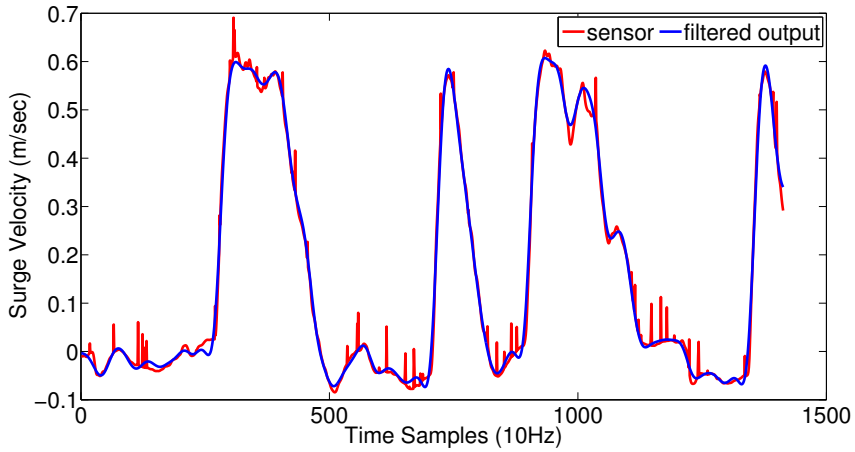


Figure 5.12: Comparison between the sensor provided to the algorithm and the yielded state. Obviously the system is able to reject outliers without considering that particular situation as a possible dynamic shift.

5.4 Remarks

The presented algorithm is able to detect faults in real-time without any prior knowledge of the fault dynamics. Moreover, the dynamic representation of the autonomous system is adapted, providing useful information for either control or planning. The experimental evaluation supports the validity of the algorithm. Furthermore, the robustness to sensor outliers has been preserved. The presented framework is versatile, for it can be applied to any autonomous system. Knowledge about possible faults or the faults frequency can be incorporated by either adding a new dynamic model to the Mixture of Gaussians or by adjusting the prior probability respectively.

Being able to distinguish sensor outliers from dynamic alterations increases significantly the robustness of the system. Moreover, the algorithm learns the new condition (fault or dynamic alteration more generally). The new model can then be incorporated to any standard control scheme (jacobian inverse, jacobian transpose) to achieve the desired behaviour, despite the dynamic abnormality. Using a maximum-likelihood approach, the algorithm fine-tunes the required parameters in real-time from incoming data from the sensors; however, an initial guess needs to be provided by the user. The initial guess for the parameters is provided in the form of prior distribution.

Chapter 6

Prognostics

6.1 Introduction

Apart from dealing with sensor failures and dynamic alteration, it is often important to predict the *Remaining Useful Life* (RUL) of a system (asset). RUL is defined as the time interval during which the asset's performance satisfies certain qualitative criteria. Predicting the RUL of an autonomous system is the subject of *Prognostics*. Prognostics can be seen as an extrapolation of fault detection schemes into the future (see Figure 6.1). The *prognostic horizon* defines how far into the future a prognostic system can produce meaningful predictions. By understanding the factors that are relevant to the RUL, the operation and maintenance of the asset can be optimised. Furthermore, the insight provided by the prognostic models informs Design for Reliability (DFR) for next generation assets.

Modern autonomous systems are equipped with a variety of sensors; each monitoring part of the system's state. As a result, historical operational data from real-world systems are abundant. Moreover, machine learning has evolved sufficiently to provide a plethora of analytical tools. The latter deal effectively with high dimensional data; abstracting the important information, helping as such the user to extract the meaningful part of the data. Naturally, prognostics have also pursued the trend of data-driven algorithms.

There are two important issues, however, that need to be addressed to enable successful prognoses; i.e., correct prediction of the Remaining Useful Life. Firstly,

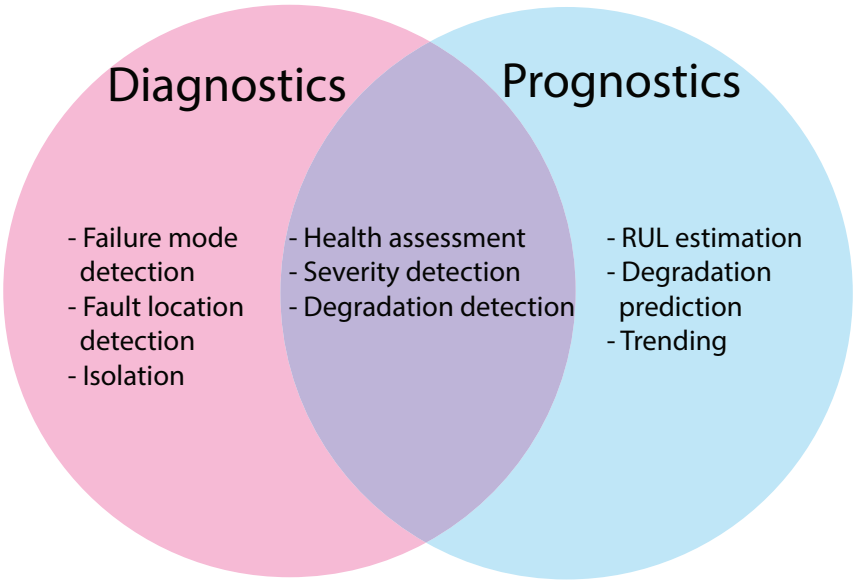


Figure 6.1: Schematic representation of diagnostics and prognostics overlap

the available data are not always suitable for training machine learning algorithms. System operation is often terminated long before complete failure. Consequently, the available data are censored; the full state trajectory is not recorded until failure. Even in the case of uncensored data, however, the majority of the data samples belong to the healthy region of operation. As a result, healthy and faulty data samples are not equally represented in the dataset; leading to a *skewed* dataset. Skewness in the data hinders algorithm training. Figure 6.2 depicts the case of a skewed dataset. Specifically, in this dataset the red and blue classes are equally represented by 300 data samples; whereas, the green class contributes only 10 samples to the dataset. Traditional classification algorithms prefer the samples of the majority class; since, the latter carry higher misclassification penalties. To remedy that, different weights can be used for each class when calculating the misclassification error. A more powerful solution combines the strength of *bagging* and *boosting* (see Appendix B for a brief explanation of the two terms) into a robust classifier for the case of imbalanced set. The indicated classifier is described later in the chapter (see Section 6.3). Another common problem pertains to the system's dynamic variance. Different operating conditions, as well as manufacturing uncertainty are two possible causes of such a variance. The need for a "smart" model, which is robust to dynamic versatility, is profound for prognostics.

In this chapter, a novel prognostic algorithm is presented, which addresses both

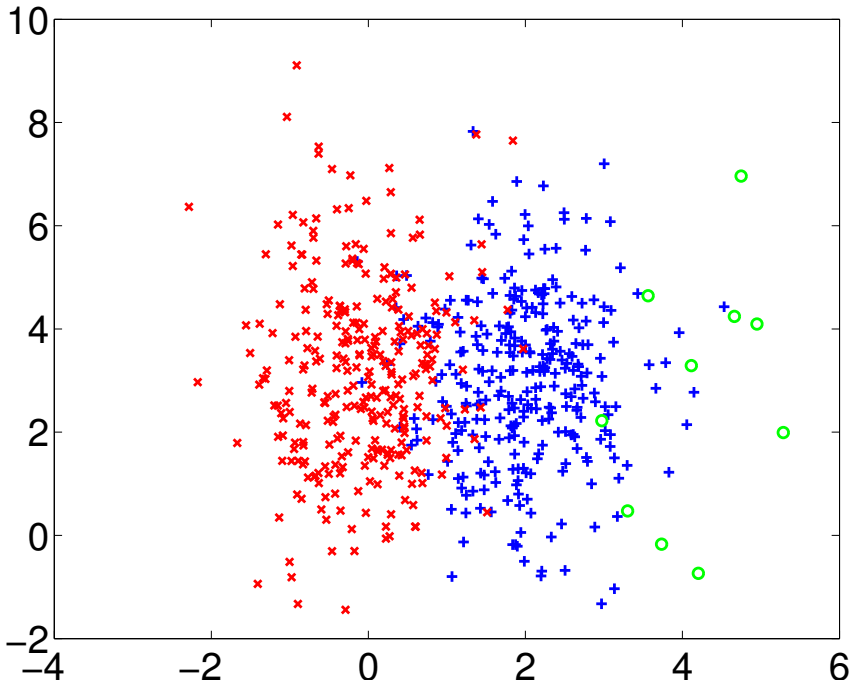


Figure 6.2: Synthetic example of a skewed dataset within a classification problem

the problems of data skewness and dynamic variance. The suggested algorithm employs a state-of-the-art classifier, together with an adaptive autoregressive predictive model. Random Undersampling Boosting (RUSBoost) [177], is used to classify the state of the system as healthy or faulty; hence acting as a fault detector. RUSBoost addresses the skewness of the dataset using random undersampling of the over-represented class. Moreover, RUSBoost can be applied without loss of performance on multi-class problems. This is particularly useful in the presence of multiple types of faults. Similarly to the previous chapters, the required dynamic model is based on Locally Weighted Projection Regression (LWPR) [176], a regression algorithm for non-linear functions with high dimensional input. The dynamic model is utilised to forecast the evolution of the system's state. The predicted state trajectory, as computed by the dynamic model, is fed to the classifier. The classifier, in turn, infers the health state of the asset. The first time instant at which the state is classified as faulty is returned by the algorithm as the remaining useful life of the asset.

The presented prognostic algorithm requires labelled data recorded at the time of failure. Unfortunately, such data were not available for the case of underwater navigation. For this reason, the algorithm was tested on two datasets, as provided from NASA's Ames Research Centre [178, 179]. The data from NASA are simulated;

however, complex models have been employed to make the yielded data realistic. The dataset comprises historical data from a fleet of one hundred simulated turbofan engines.

6.2 Adaptive Autoregression

The prognostic accuracy depends on the reliable prediction of the system's state. Moreover, the system's model needs to be adaptive, to account for variance in dynamic parameters, as well as for diverse operating conditions. To accomplish that, LWPR [176] has been used. In the rest of this section, the adaptive autoregression part of the prognostic approach is described.

The prognostic algorithm uses LWPR to predict the asset's future state, given part of the state history (depending on the order of the model) and the current input (see Equation 6.1).

$$y_n = f(y_{n-k:n-1}, u_{n-m:n-1}) \tag{6.1}$$

Equation 6.1 describes the general case of an Auto-Regressive model with Exogenous inputs (ARX). In typical ARX models $f(.,.)$ is a polynomial of the inputs and the previous states of the system: $A(B)y(t) = C(B)u(t-1)$, where A and C are polynomials of the *back shift operator* B. LWPR extends the representational power of the ARX, to include arbitrarily complex functions of the inputs. However, using LWPR doesn't yield theoretical guarantees about the ARX model's stability. The authors intend to investigate further this matter.

The autoregressive model, based on LWPR, was trained as described in section 3.3.2. The optimal initial width for the model's receptive field was chosen to be 40 (unit's vary with input dimension - hence they are omitted), the initial learning rate for the gradient descent was set to be equal to 20, whereas the penalty for infinitely small receptive fields (regularisation term) was fixed to 0.001.

Figures 6.3, 6.4 show the generalisation capacity of the resulting model. The performance of the model is tested on a dataset, which was not used for training. In

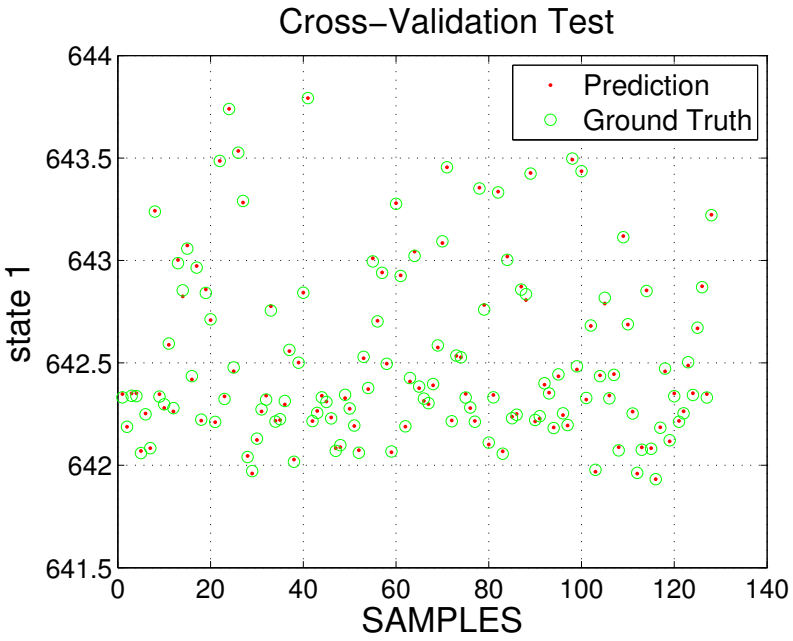


Figure 6.3: This graph shows the one step prediction from the model on a cross-validation set compared with the original value. The size of the cross-validation set consists of approximately 20% of the full data. The mean square error for the one-step prediction was $1.1 \cdot 10^{-4}$.

Figure 6.3, the model was utilised to perform one-step predictions on random state samples. In Figure 6.4, the estimated trajectory of the state is compared with the ground truth.

6.3 Robust Classification

State-of-the-art classification algorithms combine undersampling or oversampling with boosting. In boosting, multiple weak classifiers are trained sequentially, to compensate for the misclassified samples. The most popular algorithm of this class is AdaBoost [180]. AdaBoost begins with the training of a single weak classifier (e.g., SVM). The misclassified samples of the training dataset are given higher weights. Then another weak classifier is trained on the dataset, with focus on the data with higher weights. This process is repeated until the desired performance has been achieved. Below, two algorithms, which utilise sampling together with boosting, are described.

Synthetic Minority Oversampling TEechnique Boosting (SMOTEBoost) [181] iteratively trains classifiers on datasets with oversampled instances of the under-

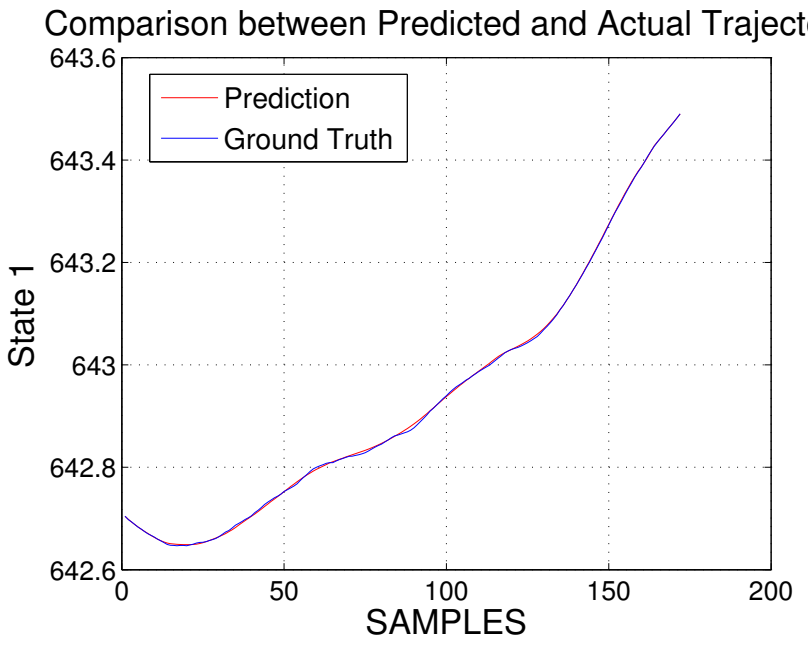


Figure 6.4: This graph shows the reconstructed trajectory of the first state of the engine as computed by our model. The auto-regressive model uses 8 lags for both the input and the output. To create this graph the actual input of the engine was used. We can see that the estimated trajectory accurately follows the original state trajectory of the engine.

represented class [182]. In this way, the importance of the misclassification for the minority class is increased. The overfitting problem persists, albeit the misclassification error is decreased.

Random Under-Sampling Boosting (RUSBoost) [177] balances the training set by removing instances of the over-represented class. One weak classifier is trained on the resulting dataset. Then the classifiers performance is checked against the original dataset. The weights are adjusted as in the AdaBoost algorithm to give higher importance to the misclassified instances. A new dataset is created and the process is repeated. In this way, RUSBoost avoids the drawbacks of discarding useful information, while it preserves the class balance within each dataset.

RUSBoost was applied on the training dataset, provided by NASA Ames Center. Decision Trees were the weak classifier of choice. Figure 6.5 shows the misclassification rate as a function of the number of weak classifiers used. The confusion matrix of the RUSBoost classifier and of an AdaBoost classifier are shown in Table 6.1 for comparison. It is obvious how RUSBoost outperforms AdaBoost, especially in the number of false positives.

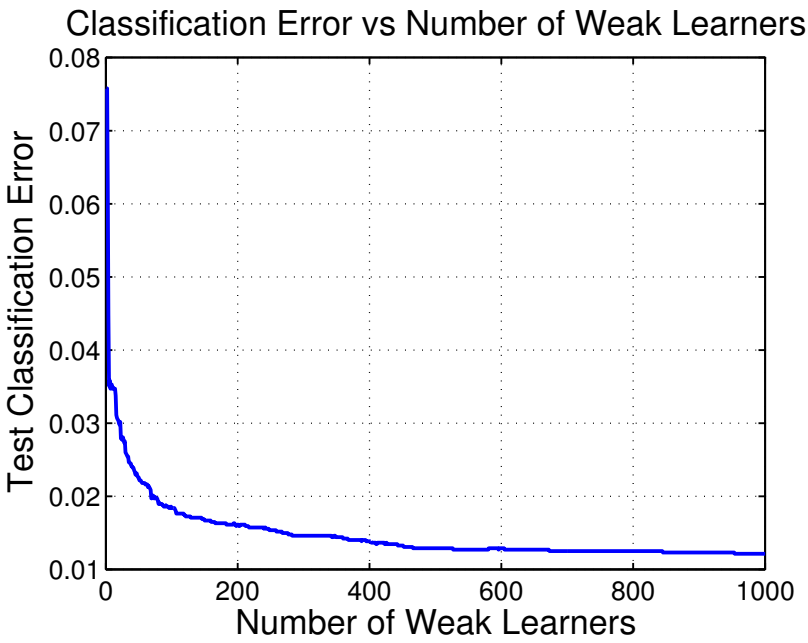


Figure 6.5: This figure shows the miss-classification rate for the RUSBoost classifier versus the number of weak learners used. The performance of the algorithm was evaluated on a cross-validation set. The classification error converges asymptotically to its minimum after 700 weak learners.

Table 6.1: Confusion matrices for AdaBoost (above) and RUSBoost (below)

		Predicted	
Actual	Class	Healthy	Faulty
	Healthy	100	0
	Faulty	100	0

		Predicted	
Actual	Class	Healthy	Faulty
	Healthy	98.8073	1.1927
	Faulty	0	100

6.4 Remaining Useful Life Computation

The remaining useful life is computed by combining the autoregressive model with the classifier. Firstly, the LWPR is trained on a subset of the full dataset. After the training is over, the algorithm is equipped with an ensemble of $n \times m$ models, where n is the number of engines in the training dataset and m is the dimension of the engine's state space. The RUSBoost classifier is trained on the same dataset. Following that, for each engine in the test dataset a set of $1 \times m$ models is selected from the ensemble, which most accurately capture its dynamics. To this end, the response of all the models in the ensemble is simulated, using the input of the engine of interest. Then for each estimated trajectory the *coefficient of determination* (see Equation 6.2) is computed. The model with a coefficient nearest to unity is selected.

$$\begin{aligned} R^2 &= 1 - \frac{SS_{res}}{SS_{tot}} \\ SS_{res} &= \sum_i (y_i - f_i)^2 \\ SS_{tot} &= \sum_i (y_i - \bar{y})^2 \end{aligned} \tag{6.2}$$

After each engine has been associated with a model, the respective states are simulated and fed into the classifier. The time instant at which the state is classified as faulty is the remaining useful life of the engine. A major issue is that the future input of the engine is not known a priori. A simple solution would be to use a constant input, equal to the mean of the input hitherto. Alternatively, the input could be kept constant to the value of the *Root Mean Square* (RMS) or to its last known value. A more sophisticated way to deal with this problem would be to use a separate AR model to forecast the input. This model is trained using the input history. During the experiments, all the aforementioned input forecasting techniques were tested. The results are discussed in detail in section 6.5.

6.5 Experiments

The dataset that was used to test the prognostic algorithm comprises information from one hundred turbofan engines. For each engine, data samples from twenty-one sensors, together with three input variables (operating conditions) have been recorded. Different initial wear and variation in the engine's dynamics have been assumed. The engine develops a fault at a random time instant. The state trajectories, however, are recorded from the beginning of the engine's operation, to the point of failure. Only one fault is present in this dataset. We used 80% of the engines for training and 20% for validation. The classifier and the adaptive autoregressive models were trained as described above.

For each engine in the validation set, a model was selected, as discussed in Section 6.4. Next, the model was simulated starting from a user-defined time (e.g. 50 time samples before the end of the trajectory) to the future. The starting time of the simulation may influence the prognostic accuracy, depending on which input forecasting method is used. In Figure 6.6 the total life of the test engines is shown. In this experiment, the future input is assumed to be known a priori. The algorithm predicts the Remaining Useful Life with a mean square error of six cycles. Next, in an effort to relax the assumption with respect to the input, three more experiments have been conducted. In the first experiment, the input was kept constant, to its latest value (see Figure 6.7). In the next experiment, the input was kept constant to the mean of the input history (see Figure 6.8). Finally, Figure 6.9 shows the total life prediction, when the input was forecasted by a fourth-order auto-regressive model.

One last experiment was performed, to test the robustness of the RUSBoost algorithm, in the case of multiple faults. To this end, the prognostic algorithm was applied on a dataset with two types of faults. The results are illustrated in Figure 6.10. The performance of the classifier was not influenced by the presence of different fault types.

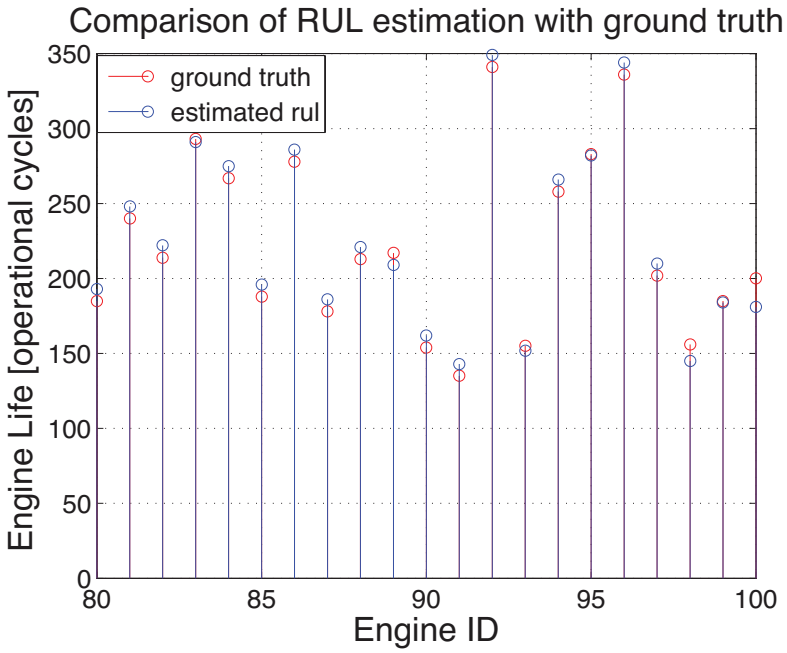


Figure 6.6: This graph shows the total engine life, as given by the dataset, compared to the total life, computed by our prognostic algorithm. The total life is computed by adding the time when the forecast started with the remaining useful life prediction of the engine. The input used in this experiment was assumed to be known (the input history from the dataset has been used)

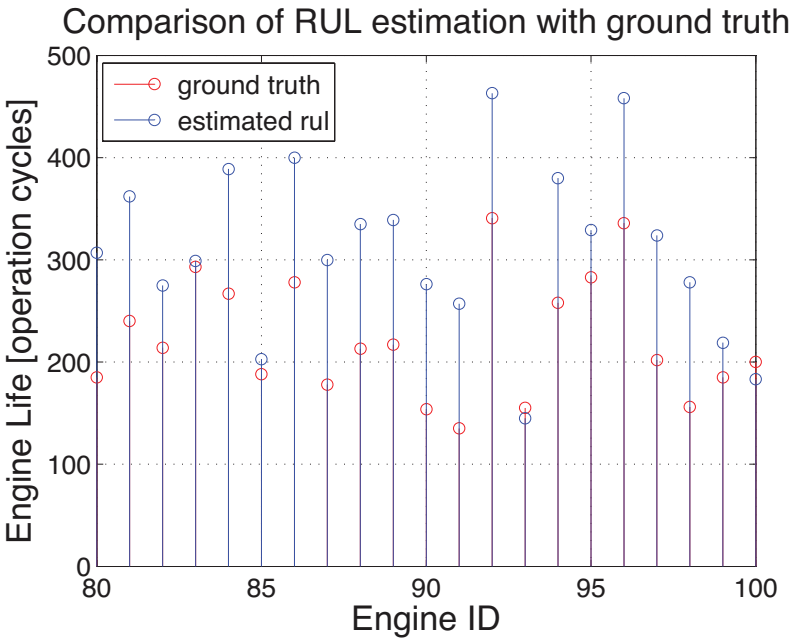


Figure 6.7: In this experiment, the computation was repeated as described in Figure 6.6 with different assumptions over the input. Here we didn't use the true values of the future input. A constant value equal to the last known input was used for the future predictions of the model. We can see that the accuracy of the estimation dropped significantly.

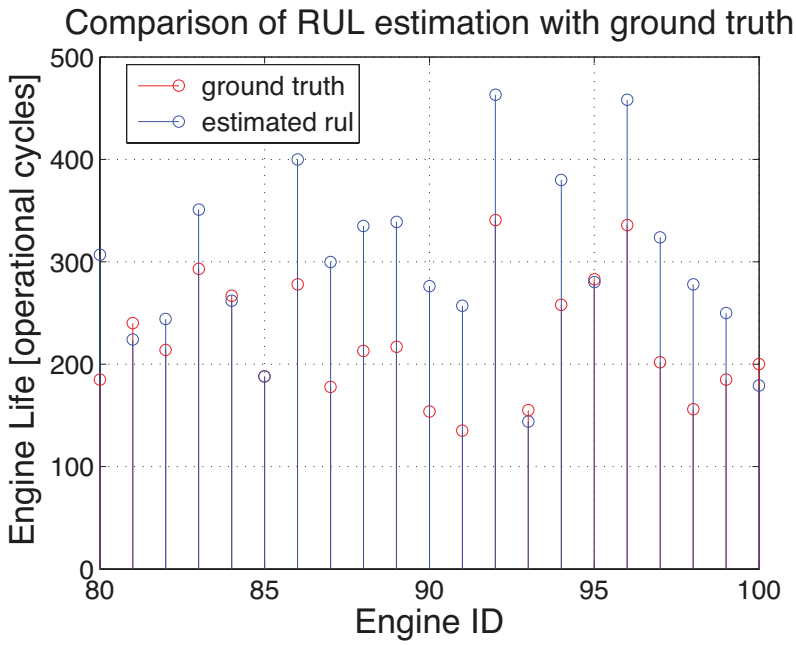


Figure 6.8: In this experiment, the input was assumed to be equal to the mean value of the previous input trajectories. Again, the performance becomes worse for the majority of the test engines.

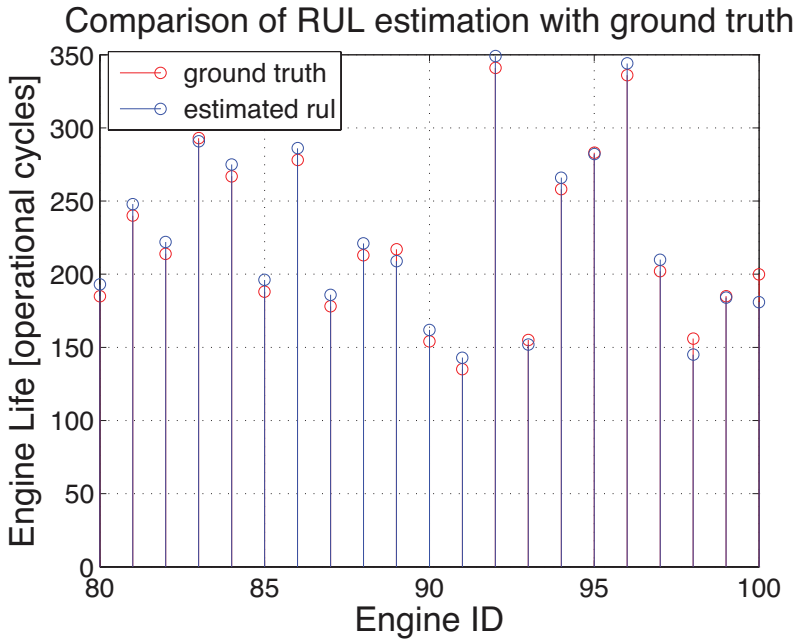


Figure 6.9: In this experiment, the input was forecasted by an Auto-Regressive model (AR). The AR model was of order 4 and its parameters were identified using the past input history. In this figure, we can see that the algorithm still predicts the total life of the engine quite accurately.

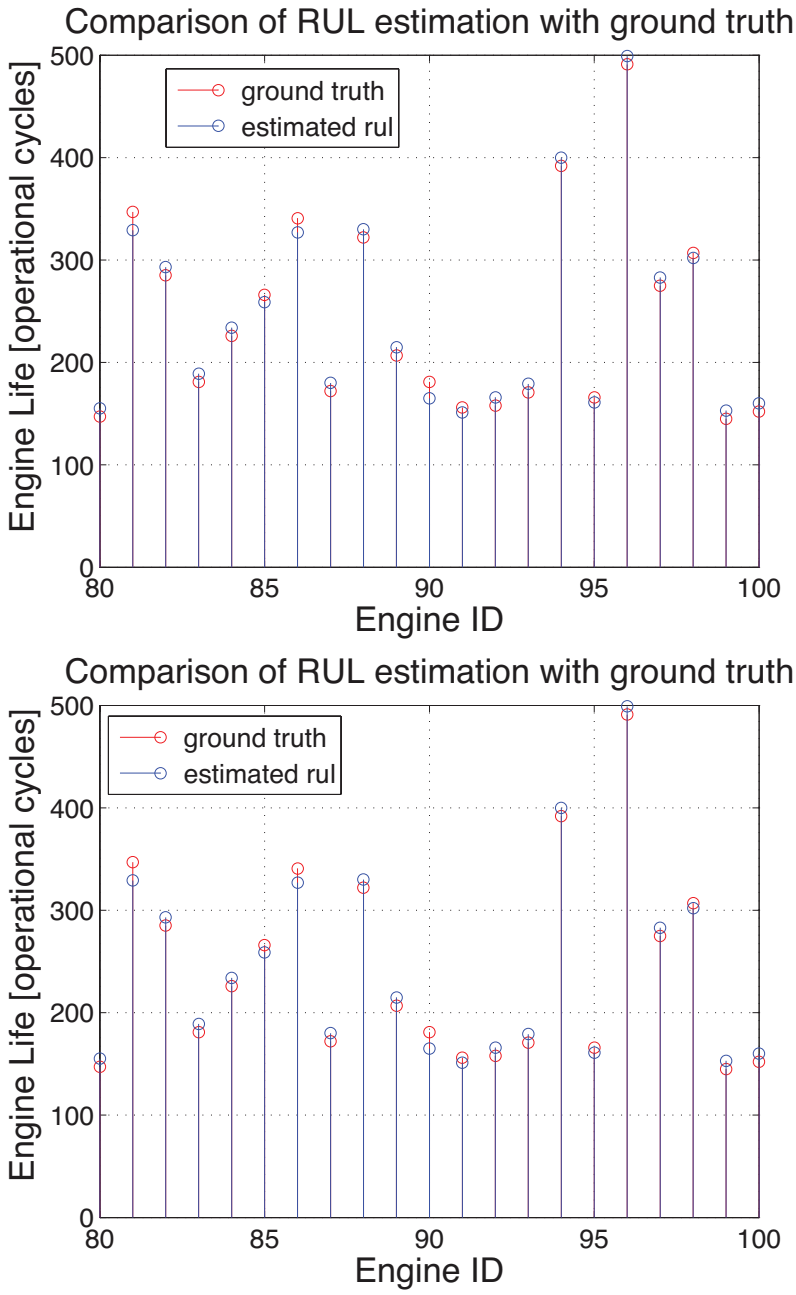


Figure 6.10: We repeated the experiment described in Figure 6.6 on another dataset provided by Ames Centre. The difference in that one is that it comprises engines with two types of fault. The purpose of this experiment was to check the robustness of the RUSBoost classifier in the case of multiple faults. The figure above shows the results when the original input is used. The figure below was created using an autoregressive model to forecast the input.

6.6 Remarks

In this chapter, a novel approach for the computation of the Remaining Useful Life was presented. Combining a state of the art regression method for state estimation, together with a robust classifier yielded accurate prognostic predictions. In the case of unknown future inputs, an Auto-Regressive model has been used. Forecasting the input using AR doesn't seem to influence the accuracy of the predictions. Moreover, the variation of the engine's dynamics has been addressed. Given a representative training dataset, it was shown that the algorithm is robust to manufacturing uncertainty. Additionally, the algorithm performed equally well on the dataset where two distinctive faults occurred.

Further investigation concerning the theoretical stability of the autoregressive model is required. Moreover, the model selection process can be improved. Initial wear may be introduced as a latent variable (i.e., inferred by the observed data at runtime) within the predictive model. In this manner, the model selection will be more rigorous and computationally efficient.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The purpose of this thesis was to increase the robustness of autonomous systems, by bridging the gap between machine learning and systems engineering. Specifically, the main focus revolved around hardware reliability; providing algorithms that can predict, identify and mitigate hardware failure. State-of-the-art regression, classification and reasoning frameworks have been combined to achieve this goal.

Nessie, an Autonomous Underwater Vehicle, was used to test most algorithm presented in this thesis. Hardware malfunctioning is common in AUVs; offering an opportunity for benchmarking hardware mitigation algorithms. Artificial sensor and actuator failures were straightforward to introduce. Practical considerations prevented a test of the prognostic algorithm on real-world data. The prognostic algorithm requires large historical datasets that contain information about the system at the moment of hardware failure. Such datasets were not available for the AUV at hand. Data sharing and privacy considerations prevented a real-world test on an industrial asset.

In real applications, the environment of operation changes continuously. Dynamic models, which are used to predict how the world evolves, need to adjust accordingly. Moreover, uncertainty of all sorts aggravates predicting the world's state. To achieve autonomous behaviour, both the environment's volatility and uncertainty need to be addressed. Persistent autonomy entails adaptivity to changes

in the environment, combined with sophisticated inference mechanisms to alleviate uncertainty.

In this thesis, the world model was solely based on Locally Weighted Projection Regression. LWPR is powerful in representing complex relations. As an incremental algorithm, LWPR endows the model with adaptive traits. To the best of our knowledge, we were the first to apply LWPR at the underwater domain [156]; highlighting the algorithm's potential and studying the requirements tailored to the needs of underwater navigation. By combining local linear models, LWPR affords straightforward Jacobian computation, which is quintessential for robotic applications and autonomous systems in general.

Sensor fault mitigation was a central topic of this thesis. Defective sensors cause data blackouts, outlier contamination of the measurement stream and frozen output values. Bayesian Reasoning has been employed to detect and, in turn, reject corrupted measurements off the sensor stream. The previous work of [1] has been extended to consider systems with nonlinear dynamics. Sensor outlier rejection was studied extensively on real-world data; the data were recorded using the in-house AUV Nessie [183]. Experimental validation highlights the algorithm's potential for navigation. Sophisticated dynamic modelling enabled robustness in frozen sensor identification [184]; compared to just occasional random outlier detection presented in previous work. The generality of the approach permits seamless integration with existing techniques, to yield a reliable navigation algorithm. Prediction of model confidence together with Bayesian treatment of sensor updating yields a self-tuning navigation algorithm.

Next, failures that change the system's dynamics have been investigated. Specifically, the use case was a defective thruster in Nessie's actuation system. The previous Bayesian representation was extended to consider simultaneously multiple explanations [185]. Each explanation is a probability distribution across the robot's state; an LWPR model predicts the mean of each distribution. A Mixture of Gaussians constitutes the substrate for combining multiple distributions. The resulting algorithm detects hardware faults during operation. Moreover, it learns a new representation

for the system's dynamics, accounting for the hardware failure. Disambiguating between a sensor defect and a model shift is hard. The transition between normal operation and failure, both sensor and actuator, provides the required information. When a sensor fails, its output value changes instantly; whereas, the effects of dynamic shifts are slower due to inertial phenomena. Prior distributions modulate the algorithm's response to such differences. Similar to sensor fault detection, it is possible to choose prior distribution parameters autonomously. However, time constraints prevented implementation of this feature in this project.

Further to fault detection and mitigation, a prognostic algorithm has been presented. The suggested algorithm [186] combines adaptive autoregression with RUSBoost to compensate for imbalanced data; i.e., for the underrepresentation of samples from when the fault occurs, compared to healthy ones. Lack of data hindered experimental evaluation; hence, this algorithm has been tested only in simulation. Using LWPR and Robust Classification, the algorithm effectively addresses the high dimensionality of prognostic data and dynamic variability, while in parallel solves the problem of data skewness.

7.2 Future Work

In this thesis, the dynamic model has been used merely for state estimation. In the AUV experiments, this corresponds to navigation. An accurate dynamic model, however, can be further utilised. Controllers leverage information about the system to predict the command that brings the system to a particular state. Most commonly, this is accomplished using the model as a *feedforward* term. Other Jacobian-based control schemes are also applicable in this case.

Bayesian reasoning proved effective in dealing with outliers. This particular algorithm is readily applicable to a broad range of autonomous systems. Nevertheless, the counter play between similar sensors have not been studied; albeit most probably sensor redundancy will make the problem even easier. Sensor redundancy will be very useful to distinguish between sensor failures and changes in the systems dynamics. Low-level fault detection, as presented in this thesis, may be also combined with

high-level diagnostic mechanisms. Instead of attempting to infer hardware integrity by merely monitoring the state, information from hardware specific diagnostic modules may be also included in the inference.

The prognostics algorithm presented here exploits large amounts of data and addresses both dynamic variability and data skewness. Nevertheless, more in-depth testing is required on a real platform. Model selection and adaptation can benefit from a Bayesian perspective. High-level decision integration would be of particular interest as well.

An AUV was the system of choice, to demonstrate most of the algorithms within this thesis. However, such algorithms can improve the performance of a broad range of autonomous systems. Navigation is fundamental in robotics; hence, the presented algorithm may prove of particular use in areas such as aerial and field robotics. Even more generally, adaptive dynamic modelling and sensor fault detection may be applied to any system as is; except that the model will need to be trained again to capture the dynamics of the system. It would be interesting to stress the presented framework in different environments, testing as such the robustness and revealing potential limitations.

Adaptive modelling has provided a robust representation of the system dynamics. Combined with Bayesian reasoning, detection of both sensor and hardware faults became feasible. Moreover, the adaptive traits of the dynamic system provide further possibilities; enabling the online learning of a dynamic model that considers the hardware defect. This adapted model can be exploited to react to hardware failure. Specifically, the new dynamic model can be used within a control framework to attempt mission completion under the new circumstances. The straightforward computation of the model's Jacobian allows the application of standard control schemes from the literature. The Jacobian of the adapted model will compensate for the change due to the hardware failure. In this manner, the autonomous system will be endowed with a fast reaction mechanism; one that mitigates, if possible, the hardware failure.

In the case of significant changes, simple jacobian-based control schemes may

not suffice. Careful consideration of the original task requirements may be necessary. For example, a thruster failure in an AUV's actuation system may impair locomotion. Again, the system's Jacobian can be used to remedy this situation. By computation of the manipulability ellipsoid, the decision maker can figure out along which dimensions it is easier for the AUV to move. A motion planning algorithm can then be used to search for an alternative plan; favouring the directions indicated by the manipulability ellipsoid. When the vehicle loses a degree of freedom, non-holonomic planners may be utilised.

Bibliography

[1] J.-A. Ting, E. Theodorou, and S. Schaal, "A kalman filter for robust outlier detection," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 1514–1519, IEEE, 2007.

[2] K. Posse, A. Crouch, J. Rearick, B. Eklow, M. Laisne, B. Bennetts, J. Doege, M. Ricchetti, and J. Cote, "Ieee p1687: toward standardized access of embedded instrumentation," in *Test Conference, 2006. ITC'06. IEEE International*, pp. 1–8, IEEE, 2006.

[3] A. K. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing*, vol. 20, pp. 1483–1510, Oct. 2006.

[4] G. Dalpiaz, A. Rivola, and R. Rubini, "Effectiveness and sensitivity of vibration processing techniques for local fault detection in gears," *Mechanical Systems and Signal Processing*, vol. 14, no. 3, pp. 387–412, 2000.

[5] A. J. Miller, *A New Wavelet Basis For The Decompostion Of Gear Motion Error Signals And Its Application To Gearbox Diagnostics.* PhD thesis, The Pennsylvania State University, 1999.

[6] S. Pöyhönen, P. Jover, and H. Hyötyniemi, "Signal processing of vibrations for condition monitoring of an induction motor," in *Control, Communications and Signal Processing, 2004. First International Symposium on*, pp. 499–502, IEEE, 2004.

[7] D. Baillie and J. Mathew, "A comparison of autoregressive modeling tech-

niques for fault diagnosis of rolling element bearings," *Mechanical Systems and Signal Processing*, vol. 10, no. 1, pp. 1–17, 1996.

[8] A. Garga, B. Elverson, and D. Lang, "Ar modeling with dimension reduction for machinery fault classification," *Critical Link: Diagnosis to Prognosis, Haymarket*, pp. 299–308, 1997.

[9] Y. Zhan, V. Makis, and A. Jardine, "Adaptive model for vibration monitoring of rotating machinery subject to random deterioration," *Journal of Quality in Maintenance Engineering*, vol. 9, no. 4, pp. 351–375, 2003.

[10] J. R. Stack, R. G. Harley, and T. G. Habetler, "An amplitude modulation detector for fault diagnosis in rolling element bearings," *Industrial Electronics, IEEE Transactions on*, vol. 51, no. 5, pp. 1097–1102, 2004.

[11] G. W. Blankenship and R. Singh, "Analytical solution for modulation sidebands associated with a class of mechanical oscillators," *Journal of Sound and Vibration*, vol. 179, no. 1, pp. 13–36, 1995.

[12] D. Ho and R. Randall, "Optimisation of bearing diagnostic techniques using simulated and actual bearing fault signals," *Mechanical systems and signal processing*, vol. 14, no. 5, pp. 763–788, 2000.

[13] M. A. Minnicino II and H. J. Sommer III, "Detecting and quantifying friction nonlinearity using the hilbert transform," in *NDE for Health Monitoring and Diagnostics*, pp. 419–427, International Society for Optics and Photonics, 2004.

[14] N. Van der Merwe and A. Hoffman, "A modified cepstrum analysis applied to vibrational signals," in *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, vol. 2, pp. 873–876, IEEE, 2002.

[15] C.-C. Wang and G.-P. J. Too, "Rotating machine fault detection based on hos and artificial neural networks," *Journal of intelligent manufacturing*, vol. 13, no. 4, pp. 283–293, 2002.

[16] X. Liangcai, S. Tielin, Y. Shuzi, and R. B. Rao, "A novel application of wavelet-based bispectrum analysis to diagnose faults in gears," *International Journal of COMADEM, vol. 5, iss. No. 3, p. 31-38*, vol. 5, pp. 31–38, 2002.

[17] D.-M. Yang, A. Stronach, P. MacConnell, and J. Penman, "Third-order spectral techniques for the diagnosis of motor bearing condition using artificial neural networks," *Mechanical systems and signal processing*, vol. 16, no. 2, pp. 391–411, 2002.

[18] B. E. Parker, H. A. Ware, D. P. Wipf, W. R. Tompkins, B. R. Clark, E. C. Larson, and H. V. Poor, "Fault diagnostics using statistical change detection in the bispectral domain," *Mechanical systems and signal processing*, vol. 14, no. 4, pp. 561–570, 2000.

[19] T. Chow and G. Fei, "Three phase induction machines asymmetrical faults identification using bispectrum," *Energy Conversion, IEEE Transactions on*, vol. 10, no. 4, pp. 688–693, 1995.

[20] N. Arthur and J. Penman, "Inverter fed induction machine condition monitoring using the bispectrum," in *Higher-Order Statistics, 1997., Proceedings of the IEEE Signal Processing Workshop on*, pp. 67–71, IEEE, 1997.

[21] A. McCormick and A. K. Nandi, "Bispectral and trispectral features for machine condition diagnosis," in *Vision, Image and Signal Processing, IEE Proceedings-*, vol. 146, pp. 229–234, IET, 1999.

[22] L. Qu, X. Liu, G. Peyronne, and Y. Chen, "The holospectrum: a new method for rotor surveillance and diagnosis," *Mechanical Systems and Signal Processing*, vol. 3, no. 3, pp. 255–267, 1989.

[23] Q. L. S. Dongfeng, "Holospectrum during the past decade: Review & prospect [j]," *Journal of Vibration, Measurement & Diagnosis*, vol. 4, p. 000, 1998.

[24] W. Wang and P. McFadden, "Early detection of gear failure by vibration analysis i. calculation of the time-frequency distribution," *Mechanical Systems and Signal Processing*, vol. 7, no. 3, pp. 193–203, 1993.

[25] W. Staszewski and G. Tomlinson, "Application of the wavelet transform to fault detection in a spur gear," *Mechanical Systems and Signal Processing*, vol. 8, no. 3, pp. 289–307, 1994.

[26] R. Rubini and U. Meneghetti, "Application of the envelope and wavelet transform analyses for the diagnosis of incipient faults in ball bearings," *Mechanical systems and signal processing*, vol. 15, no. 2, pp. 287–302, 2001.

[27] N. Aretakis and K. Mathioudakis, "Wavelet analysis for gas turbine fault diagnostics," *Journal of engineering for gas turbines and power*, vol. 119, no. 4, pp. 870–876, 1997.

[28] G. Dalpiaz and A. Rivola, "Condition monitoring and diagnostics in automatic machines: comparison of vibration analysis techniques," *Mechanical Systems and Signal Processing*, vol. 11, no. 1, pp. 53–73, 1997.

[29] P. Addison, J. Watson, and T. FENG, "Low-oscillation complex wavelets," *Journal of Sound and Vibration*, vol. 254, no. 4, pp. 733–762, 2002.

[30] X. Yin-ge and Y. Yu-ling, "Research on haar spectrum in fault diagnosis of rotating machinery," *Applied Mathematics and Mechanics*, vol. 12, no. 1, pp. 61–66, 1991.

[31] C. Wang and R. X. Gao, "Wavelet transform with spectral post-processing for enhanced feature extraction [machine condition monitoring]," *Instrumentation and Measurement, IEEE Transactions on*, vol. 52, no. 4, pp. 1296–1301, 2003.

[32] G. G. Yen and K.-C. Lin, "Wavelet packet feature extraction for vibration monitoring," *Industrial Electronics, IEEE Transactions on*, vol. 47, no. 3, pp. 650–667, 2000.

[33] H. Yang, J. Mathew, and L. Ma, "Fault diagnosis of rolling element bearings using basis pursuit," *Mechanical Systems and Signal Processing*, vol. 19, no. 2, pp. 341–356, 2005.

[34] Z. Peng and F. Chu, "Application of the wavelet transform in machine condition monitoring and fault diagnostics: a review with bibliography," *Mechanical systems and signal processing*, vol. 18, no. 2, pp. 199–221, 2004.

[35] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE Transactions on Control Systems Technology*, vol. 18, pp. 636–653, May 2010.

[36] M. O. Cordier, P. Dague, F. Levy, J. Montmain, M. Staroswiecki, and L. Trave-Massuyes, "Conflicts versus analytical redundancy relations," *The special issues of the IEEE SMC Transactions-Part B on Diagnosis of Complex Systems: Bridging the methodologies of the FDI and DX Communities*, 2003.

[37] J. Gertler, "Fault detection and isolation using parity relations," *Control engineering practice*, vol. 5, no. 5, pp. 653–661, 1997.

[38] R. Patton and J. Chen, "Robust fault detection using eigenstructure assignment: a tutorial consideration and some new results," in *Decision and Control, 1991., Proceedings of the 30th IEEE Conference on*, pp. 2242–2247, IEEE, 1991.

[39] R. Patton and J. Chen, "A review of parity space approaches to fault diagnosis," in *IFAC Safeprocess Conference*, pp. 65–81, 1991.

[40] B. O. Bouamama, A. Samantaray, M. Staroswiecki, and G. Dauphin-Tanguy, "Derivation of constraint relations from bond graph models for fault detection and isolation," *SIMULATION SERIES*, vol. 35, no. 2, pp. 104–109, 2003.

[41] J. Thoma, *Introduction to bond graphs and their applications*. Pergamon international library of science, technology, engineering, and social studies, 1975., 1975.

[42] P. J. Mosterman and G. Biswas, "A theory of discontinuities in physical system models," *Journal of Franklin Institute*, vol. 335, no. B, pp. 401–439, 1998.

[43] C. B. Low, D. Wang, S. Arogeti, and J. B. Zhang, "Causality assignment and model approximation for quantitative hybrid bond graph-based fault diagnosis," in *The 17th IFAC world congress*, pp. 10522–10527, 2008.

[44] J. Gertler, "Fault detection and isolation using parity relations," *Control Engineering Practice*, vol. 5, no. 5, pp. 653–661, 1997.

[45] K. Watanabe and D. M. Himmelblau, "Instrument fault detection in systems with uncertainties," *International Journal of Systems Science*, vol. 13, no. 2, pp. 137–158, 1982.

[46] J. Wünnenberg and P. Frank, "Sensor fault detection via robust observers," in *System Fault Diagnostics, Reliability and Related Knowledge-Based Approaches* (S. Tzafestas, M. Singh, and G. Schmidt, eds.), pp. 147–160, Springer Netherlands, 1987.

[47] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part i: Quantitative model-based methods," *Computers & Chemical Engineering*, vol. 27, pp. 293–311, Mar. 2003. Cited by 1145.

[48] J. Park and G. Rizzoni, "A new interpretation of the fault detection filter part 1: Closed-form algorithm," *International Journal of Control*, vol. 60, no. 5, pp. 767–787, 1994.

[49] D. M. Wilbers and J. L. Speyer, "Detection filters for aircraft sensor and actuator faults," in *Control and Applications, 1989. Proceedings. ICCON'89. IEEE International Conference on*, pp. 81–86, IEEE, 1989.

[50] R. K. Douglas and J. L. Speyer, "Robust fault detection filter design," *Journal of guidance, control, and dynamics*, vol. 19, no. 1, pp. 214–218, 1996.

[51] R. K. Douglas and J. L. Speyer, "H bounded fault detection filter," *Journal of guidance, control, and dynamics*, vol. 22, no. 1, pp. 129–138, 1999.

[52] W. H. Chung and J. L. Speyer, "A game theoretic fault detection filter," *Automatic Control, IEEE Transactions on*, vol. 43, no. 2, pp. 143–161, 1998.

[53] J. Stoustrup and H. H Niemann, "Fault estimationa standard problem approach," *International Journal of Robust and Nonlinear Control*, vol. 12, no. 8, pp. 649–673, 2002.

[54] T. Song and E. G. Collins, "Robust h estimation with application to robust fault detection," *Journal of guidance, control, and dynamics*, vol. 23, no. 6, pp. 1067–1071, 2000.

[55] E. G. Collins Jr, W. M. Haddad, V.-S. Chellaboina, and T. Song, "Robustness analysis in the delta-domain using fixed-structure multipliers," in *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, vol. 4, pp. 3286–3291, IEEE, 1997.

[56] A. A Stoorvogel, H. H Niemann, A. Saberi, and P. Sannuti, "Optimal fault signal estimation," *International Journal of Robust and Nonlinear Control*, vol. 12, no. 8, pp. 697–727, 2002.

[57] J. S. Shamma and K.-Y. Tu, "Set-valued observers and optimal disturbance rejection," *Automatic Control, IEEE Transactions on*, vol. 44, no. 2, pp. 253–264, 1999.

[58] P. Rosa, C. Silvestre, J. S. Shamma, and M. Athans, "Fault detection and isolation of an aircraft using set-valued observers," in *Proceedings of the 18th IFAC Symposium on Automatic control in aerospace, Nara, Japan*, pp. 6–10, 2010.

[59] P. Rosa, C. Silvestre, J. S. Shamma, and M. Athans, "Multiple-model adaptive control with set-valued observers," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pp. 2441–2447, 2009.

[60] X. Zhang, M. M. Polycarpou, and T. Parisini, "A robust detection and isolation scheme for abrupt and incipient faults in nonlinear systems," *Automatic Control, IEEE Transactions on*, vol. 47, no. 4, pp. 576–593, 2002.

[61] R. K. Mehra and J. Peschon, "An innovations approach to fault detection and diagnosis in dynamic systems," *Automatica*, vol. 7, no. 5, pp. 637–640, 1971.

[62] D. T. Magill, "Optimal adaptive estimation of sampled stochastic processes," *Automatic Control, IEEE Transactions on*, vol. 10, no. 4, pp. 434–439, 1965.

[63] N. A. White, *MMAE Detection of Interference/Jamming and Spoofing in a DGPS-Aided INS*. PhD thesis, MS thesis, AFIT/GE/ENG/96D-21, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1996.

[64] T. E. Menke and P. S. Maybeck, "Multiple model adaptive estimation applied to the vista f-16 flight control system with actuator and sensor failures," in *Aerospace and Electronics Conference, 1992. NAECON 1992., Proceedings of the IEEE 1992 National*, pp. 441–448, IEEE, 1992.

[65] P. D. Hanlon and P. S. Maybeck, "Characterization of kalman filter residuals in the presence of mismodeling," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 36, no. 1, pp. 114–131, 2000.

[66] P. Hajek, "Fuzzy logic," in *The Stanford Encyclopedia of Philosophy* (E. N. Zalta, ed.), fall 2010 ed., 2010.

[67] H.-J. Zimmermann, L. A. Zadeh, and B. R. Gaines, *Fuzzy sets and decision analysis*, vol. 20. North Holland, 1984.

[68] R. Schneider and P. M. Frank, "Fuzzy logic based threshold adaption for fault detection in robots," in *Control Applications, 1994., Proceedings of the Third IEEE Conference on*, pp. 1127–1132, 1994.

[69] X. Zhang, L. Tang, and J. Decastro, "Robust fault diagnosis of aircraft en-

gines: A nonlinear adaptive estimation-based approach," *IEEE Transactions on Control Systems Technology*, 2012.

[70] J. Ma and J. C. Li, "Detection of localised defects in rolling element bearings via composite hypothesis test," *Mechanical Systems and Signal Processing*, vol. 9, no. 1, pp. 63–75, 1995.

[71] M. L. Fugate, H. Sohn, and C. R. Farrar, "Vibration-based damage detection using statistical process control," *Mechanical Systems and Signal Processing*, vol. 15, no. 4, pp. 707–721, 2001.

[72] D. P. Malladi and J. L. Speyer, "A generalized shiryayev sequential probability ratio test for change detection and isolation," *Automatic Control, IEEE Transactions on*, vol. 44, no. 8, pp. 1522–1534, 1999.

[73] E. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[74] W. A. Shewhart, *Statistical method: From the viewpoint of quality control*. DoverPublications. com, 1939.

[75] J. MacGregor and T. Kourti, "Statistical process control of multivariate processes," *Control Engineering Practice*, vol. 3, no. 3, pp. 403–414, 1995.

[76] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, no. 6, pp. 559–572, 1901.

[77] W. Ku, R. H. Storer, and C. Georgakis, "Disturbance detection and isolation by dynamic principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 30, pp. 179–196, Nov. 1995.

[78] J. Mina and C. Verde, "Fault detection using dynamic principal component analysis by average estimation," in *Electrical and Electronics Engineering, 2005 2nd International Conference on*, pp. 374–377, 2005.

[79] S. Wold, J. Trygg, A. Berglund, and H. Antti, "Some recent developments in pls modeling," *Chemometrics and Intelligent Laboratory Systems*, vol. 58, no. 2, pp. 131 – 150, 2001. ¡ce:title¿PLS Methods¡/ce:title¿.

[80] S. Yoon and J. F. MacGregor, "Statistical and causal model-based approaches to fault detection and isolation," *AIChE Journal*, vol. 46, no. 9, pp. 1813–1824, 2000.

[81] S. Verron, J. Li, and T. Tiplica, "Fault detection and isolation of faults in a multivariate process with bayesian network," *Journal of Process Control*, vol. 20, pp. 902–911, Sept. 2010.

[82] V. Skormin, L. Popyack, V. Gorodetski, M. Araiza, and J. Michel, "Applications of cluster analysis in diagnostics-related problems," in *Aerospace Conference, 1999. Proceedings. 1999 IEEE*, vol. 3, pp. 161–168, IEEE, 1999.

[83] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[84] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *The Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2002.

[85] D. Hong, G. Xiuwen, and Y. Shuzi, "An approach to state recognition and knowledge-based diagnosis for engines," *Mechanical Systems and Signal Processing*, vol. 5, no. 4, pp. 257–266, 1991.

[86] X. Lou and K. A. Loparo, "Bearing fault diagnosis based on wavelet transform and fuzzy inference," *Mechanical systems and signal processing*, vol. 18, no. 5, pp. 1077–1095, 2004.

[87] C. Bunks, D. McCarthy, and T. Al-Ani, "Condition-based maintenance of machines using hidden markov models," *Mechanical Systems and Signal Processing*, vol. 14, no. 4, pp. 597–612, 2000.

[88] J. Ying, T. Kirubarajan, K. R. Pattipati, and A. Patterson-Hine, "A hidden markov model-based algorithm for fault diagnosis with partial and imperfect tests," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 30, no. 4, pp. 463–473, 2000.

[89] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *Information Theory, IEEE Transactions on*, vol. 13, no. 2, pp. 260–269, 1967.

[90] M. J. Roemer, C.-a. Hong, and S. H. Hesler, "Machine health monitoring and life management using finite-element-based neural networks," *Journal of engineering for gas turbines and power*, vol. 118, no. 4, pp. 830–835, 1996.

[91] Y. Fan and C. J. Li, "Diagnostic rule extraction from trained feedforward neural networks," *Mechanical Systems and Signal Processing*, vol. 16, no. 6, pp. 1073–1081, 2002.

[92] K. Watanabe, S. Hirota, L. Hou, and D. Himmelblau, "Diagnosis of multiple simultaneous fault via hierarchical artificial neural networks," *AIChE Journal*, vol. 40, no. 5, pp. 839–848, 1994.

[93] J. C. Giarratano and G. Riley, *Expert systems*. PWS Publishing Co., 1998.

[94] M. F. Baig and N. Sayeed, "Model-based reasoning for fault diagnosis of twin-spool turbofans," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 212, no. 2, pp. 109–116, 1998.

[95] Z. Wen, J. Crossman, J. Cardillo, and Y. Murphey, "Case-base reasoning in vehicle fault diagnostics," in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, vol. 4, pp. 2679–2684, IEEE, 2003.

[96] R. Silva, R. Reuben, K. Baker, and S. Wilcox, "Tool wear monitoring of turning operations by neural network and expert system classification of a feature set generated from multiple sensors," *Mechanical Systems and Signal Processing*, vol. 12, no. 2, pp. 319–332, 1998.

[97] H. R. DePold and F. D. Gass, "The application of expert systems and neural networks to gas turbine prognostics and diagnostics," in *ASME 1998 International Gas Turbine and Aeroengine Congress and Exhibition*, pp. V005T15A009–V005T15A009, American Society of Mechanical Engineers, 1998.

[98] R. David and H. Alla, "Petri nets for modeling of dynamic systems: A survey," *Automatica*, vol. 30, no. 2, pp. 175–202, 1994.

[99] N. C. Propes and G. Vachtsevanos, "A fuzzy petri-net-based mode identification algorithm for fault diagnosis of complex systems," in *AeroSense 2003*, pp. 44–53, International Society for Optics and Photonics, 2003.

[100] S. Yang, "A condition-based failure-prediction and processing-scheme for preventive maintenance," *Reliability, IEEE Transactions on*, vol. 52, no. 3, pp. 373–383, 2003.

[101] B.-S. Yang, S. K. Jeong, Y.-M. Oh, and A. C. C. Tan, "Case-based reasoning system with petri nets for induction motor fault diagnosis," *Expert Systems with Applications*, vol. 27, no. 2, pp. 301–311, 2004.

[102] H. Sohn, C. Farrar, F. M. Hemez, G. Park, A. N. Robertson, and T. O. Williams, "A coupled approach to developing damage prognosis solutions," *Key Engineering Materials*, vol. 245, pp. 289–306, 2003.

[103] G. H. Ebel, "Physics of failure in commercialand," in *Physics of Failure in Electronics, 1964. Third Annual Symposium on the*, pp. 173–190, IEEE, 1964.

[104] S. Mathew, D. Das, R. Rossenberger, and M. Pecht, "Failure mechanisms based prognostics," in *Prognostics and Health Management, 2008. PHM 2008. International Conference on*, pp. 1–6, IEEE, 2008.

[105] J. R. Celaya, N. Patil, S. Saha, P. Wysocki, and K. Goebel, "Towards accelerated aging methodologies and health management of power mosfets (technical brief)," in *Annual Conference of the Prognostics and Health Management Society*, 2009.

[106] M. Pecht and J. Gu, "Physics-of-failure-based prognostics for electronic products," *Transactions of the Institute of Measurement and Control*, vol. 31, pp. 309–322, June 2009.

[107] M. Pecht, *Prognostics and health management of electronics*. Wiley Online Library, 2008.

[108] S. Mishra, M. Pecht, and D. L. Goodman, "In-situ sensors for product reliability monitoring," in *Symposium on Design, Test, Integration, and Packaging of MEMS/MOEMS 2002*, pp. 10–19, International Society for Optics and Photonics, 2002.

[109] S. Mathew, M. Osterman, and M. Pecht, "A canary device based approach for prognosis of ball grid array packages," in *Prognostics and Health Management (PHM), 2012 IEEE Conference on*, pp. 1–5, IEEE, 2012.

[110] N. Gebraeel, M. Lawley, R. Liu, and V. Parmeshwaran, "Residual life predictions from vibration-based degradation signals: a neural network approach," *Industrial Electronics, IEEE Transactions on*, vol. 51, no. 3, pp. 694–700, 2004.

[111] K. Kazmierczak, "Application of autoregressive prognostic techniques in diagnostics," in *Proceedings of the Vehicle Diagnostics Conference, Tuczno, Poland*, 1983.

[112] C. J. Lu and W. O. Meeker, "Using degradation measures to estimate a time-to-failure distribution," *Technometrics*, vol. 35, pp. 161–174, May 1993.

[113] N. Z. Gebraeel, M. A. Lawley, R. Li, and J. K. Ryan, "Residual-life distributions from component degradation signals: a bayesian approach," *IIE Transactions*, vol. 37, no. 6, pp. 543–557, 2005.

[114] D. Kumar and B. Klefsjö, "Proportional hazards model: a review," *Reliability Engineering & System Safety*, vol. 44, no. 2, pp. 177–188, 1994.

[115] N. Gorjian, L. Ma, M. Mittinty, P. Yarlagadda, and Y. Sun, "A review on degradation models in reliability analysis," in *Engineering Asset Lifecycle Management*, pp. 369–384, Springer, 2010.

[116] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, pp. 1554–1563, 1966.

[117] X.-S. Si, W. Wang, C.-H. Hu, and D.-H. Zhou, "Remaining useful life estimation a review on the statistical data driven approaches," *European Journal of Operational Research*, vol. 213, pp. 1–14, Aug. 2011.

[118] O. Geramifard, J. X. Xu, J. H. Zhou, and X. Li, "Continuous health condition monitoring: A single hidden semi-markov model approach," in *Prognostics and Health Management (PHM), 2011 IEEE Conference on*, pp. 1–10, 2011.

[119] D. A. Tobon-Mejia, K. Medjaher, N. Zerhouni, and G. Tripot, "Hidden markov models for failure diagnostic and prognostic," in *Prognostics and System Health Management Conference (PHM-Shenzhen), 2011*, pp. 1–8, 2011.

[120] D. Tobon-Mejia, K. Medjaher, N. Zerhouni, and G. Tripot, "A mixture of gaussians hidden markov model for failure diagnostic and prognostic.," in *6th Annual IEEE Conference on Automation Science and Engineering, CASE'10.*, pp. 338–343, 2010.

[121] K. P. Murphy, *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, 2002.

[122] S. Russell, "Artificial intelligence: A modern approach author: Stuart russell, peter norvig, publisher: Prentice hall pa," 2009.

[123] K. W. Przytula and A. Choi, "An implementation of prognosis with dynamic bayesian networks," in *Aerospace Conference, 2008 IEEE*, pp. 1–8, IEEE, 2008.

[124] A. Muller, M.-C. Suhner, and B. Iung, "Formalisation of a new prognosis model for supporting proactive maintenance implementation on industrial system," *Reliability Engineering & System Safety*, vol. 93, no. 2, pp. 234–253, 2008.

[125] D. Y. Kim, S.-G. Lee, and M. Jeon, "Outlier rejection methods for robust kalman filtering," in *Future Information Technology* (J. J. Park, L. T. Yang, and C. Lee, eds.), no. 184 in Communications in Computer and Information Science, pp. 316–322, Springer Berlin Heidelberg, Jan. 2011.

[126] D. C. Swanson, J. Michael Spencer, and S. H. Arzoumanian, "Prognostic modelling of crack growth in a tensioned steel band," *Mechanical systems and signal processing*, vol. 14, no. 5, pp. 789–803, 2000.

[127] M. E. Orchard and G. J. Vachtsevanos, "A particle-filtering approach for on-line fault diagnosis and failure prognosis," *Transactions of the Institute of Measurement and Control*, vol. 31, no. 3-4, pp. 221–246, 2009.

[128] J. C. Kinsey, R. M. Eustice, and L. L. Whitcomb, "A survey of underwater vehicle navigation: Recent advances and new challenges," in *IFAC Conference of Manoeuvering and Control of Marine Craft*, 2006.

[129] J. M. Hereford, "Fault-tolerant sensor systems using evolvable hardware," *Transactions on Instrumentation and Measurement*, vol. 55, no. 3, pp. 846–853, 2006.

[130] P. A. Miller, J. A. Farrell, Y. Zhao, and V. Djapic, "Autonomous underwater vehicle navigation," *Oceanic Engineering, IEEE Journal of*, vol. 35, no. 3, pp. 663–678, 2010.

[131] S. C. Martin and L. L. Whitcomb, "Preliminary experiments in comparative experimental identification of six degree-of-freedom coupled dynamic plant models for underwater robot vehicles," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 2962–2969, IEEE, 2013.

[132] M. Blain, S. Lemieux, and R. Houde, "Implementation of a rov navigation system using acoustic/doppler sensors and kalman filtering," in *OCEANS 2003. Proceedings*, vol. 3, pp. 1255–1260, IEEE, 2003.

[133] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly sparse delayed-state filters," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 2417–2424, IEEE, 2005.

[134] C. N. Roman, *Self consistent bathymetric mapping from robotic vehicles in the deep ocean*. PhD thesis, Massachusetts Institute of Technology, 2005.

[135] R. Van Der Merwe, *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. PhD thesis, University of Stellenbosch, 2004.

[136] L. Drolet, F. Michaud, and J. Côté, "Adaptable sensor fusion using multiple kalman filters.," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1434–1439, IEEE, 2000.

[137] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Factor graph based incremental smoothing in inertial navigation systems," in *Information Fusion (FUSION), 2012 15th International Conference on*, pp. 2154–2161, IEEE, 2012.

[138] H. B. Mitchell, *Multi-sensor data fusion*. Springer, 2007.

[139] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2008.

[140] M. M. Hunt, W. M. Marquet, D. A. Moller, K. R. Peal, and W. K. Smith, "An acoustic navigation system.," tech. rep., DTIC Document, 1974.

[141] P. H. Milne, *Underwater acoustic positioning systems*. Spon Press, 1983.

[142] F. P. Parthiot and J.-F. Denis, "A better way to navigate on deep sea floors," in *OCEANS'93. Engineering in Harmony with Ocean. Proceedings*, pp. II494–II498, IEEE, 1993.

[143] K. Vickery, "Acoustic positioning systems. a practical overview of current systems," in *Proceedings of the Autonomous Underwater Vehicles (AUV)*, pp. 5–17, IEEE, 1998.

[144] G. Qiao, Z. Li, Z. Sun, D. Nie, and H. Cui, "Kalman filter restraining outliers for short baseline system," in *Proceedings of the 10th World Congress on Intelligent Control and Automation (WCICA)*, pp. 322–325, IEEE, 2012.

[145] J. Vaganay, J. J. Leonard, and J. G. Bellingham, "Outlier rejection for autonomous acoustic navigation," in *IEEE Proceedings of the International Conference on Robotics and Automation (ICRA)*, vol. 3, pp. 217–2181, IEEE, 1996.

[146] J. J. Leonard, A. A. Bennett, C. M. Smith, and H. J. S. Feder, "Autonomous underwater vehicle navigation," in *IEEE ICRA Workshop on Navigation of Outdoor Autonomous Vehicles, Leuven, Belgium, May*, Citeseer, 1998.

[147] E. Wolbrecht, M. Anderson, J. Canning, D. Edwards, J. Frenzel, D. Odell, T. Bean, J. Stringfield, J. Feusi, B. Armstrong, A. Folk, and B. Crosbie, "Field testing of moving short-baseline navigation for autonomous underwater vehicles using synchronized acoustic messaging," *Journal of Field Robotics*, vol. 30, no. 4, pp. 519–535, 2013.

[148] A. Folk, B. Armstrong, E. Wolbrecht, H. F. Grip, M. Anderson, and D. Edwards, "Autonomous underwater vehicle navigation using moving baseline on a target ship," in *Proceedings of OCEANS*, pp. 1–7, IEEE, 2010.

[149] J. Vaganay, J. J. Leonard, J. A. Curcio, and J. S. Willcox, "Experimental validation of the moving long base-line navigation concept," in *IEEE/OES Autonomous Underwater Vehicles*, pp. 59–65, IEEE, 2004.

[150] J. J. Wang, W. Ding, and J. Wang, "Improving adaptive kalman filter in GPS\SDINS integration with neural network," *Proceedings of ION GNSS 2007*, 2007.

[151] Q. Song, "An adaptive ukf algorithm for the state parameter estimations of a mobile robot," *Acta Automatica Sinica*, vol. 34, May 2008.

[152] G. Agamennoni, J. I. Nieto, and E. M. Nebot, "An outlier-robust kalman filter," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1551–1558, IEEE, 2011.

[153] Z. Berman, "Outliers rejection in kalman filteringsome new observations," in *Position, Location and Navigation Symposium (PLANS)*, pp. 1008–1013, IEEE, 2014.

[154] D.-J. Jwo, C.-S. Chang, and C.-H. Lin, "Neural network aided adaptive kalman filtering for GPS applications," in *International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 3686–3691, 2004.

[155] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 742–747, Apr. 2007.

[156] G. Fagogenis, D. Flynn, and D. M. Lane, "Improving underwater vehicle navigation state estimation using locally weighted projection regression," in *International Conference on Robotics and Automation, 2014. ICRA 2014.*, IEEE, 2014.

[157] G. Antonelli, *Underwater robots motion and force control of vehicle-manipulator systems*. Berlin New York: Springer, 2006.

[158] M. Caccia, R. Bono, G. Bruzzone, G. Bruzzone, E. Spirandelli, and G. Veruggio, "Experiences on actuator fault detection, diagnosis and accomodation for rovs," *International Symposiyum of Unmanned Untethered Sub-mersible Technol*, 2001.

[159] A. Hanai, S. Choi, G. Marani, and K. Rosa, "Experimental validation of model-based thruster fault detection for underwater vehicles," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 194–199, May 2009.

[160] A. Healey, S. Rock, S. Cody, D. Miles, and J. P. Brown, "Toward an improved understanding of thruster dynamics for underwater vehicles," in *Autonomous Underwater Vehicle Technology, 1994. AUV '94., Proceedings of the 1994 Symposium on*, pp. 340–352, Jul 1994.

[161] J. Kim, J. Han, W. K. Chung, J. Yuh, and P.-M. Lee, "Accurate and practical thruster modeling for underwater vehicles," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 175–180, April 2005.

[162] S. R. Ahmadzadeh, A. Carrera, M. Leonetti, P. Kormushev, and D. G. Caldwell, "Online discovery of auv control policies to overcome thruster failures," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA 2014)*, (Hong Kong, China), June 2014.

[163] K. Yang, J. Yuh, and S. Choi, "Experimental study of fault-tolerant system design for underwater robots," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2, pp. 1051–1056 vol.2, May 1998.

[164] N. Sarkar, T. Podder, and G. Antonelli, "Fault-accommodating thruster force allocation of an auv considering thruster redundancy and saturation," *Robotics and Automation, IEEE Transactions on*, vol. 18, pp. 223–233, Apr 2002.

[165] A. Corduneanu and C. M. Bishop, "Variational bayesian model selection for mixture distributions," in *Artificial intelligence and Statistics*, vol. 2001, pp. 27–34, Morgan Kaufmann Waltham, MA, 2001.

[166] K. P. Murphy, "Switching kalman filters," tech. rep., Citeseer, 1998.

[167] S. Vijayakumar, "LWPR software tutorial (online)," 2014. http://wcms.inf.ed.ac.uk/ipab/slmc/research/software-lwpr [accessed 15-August-2014].

[168] G. E. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis: forecasting and control*. John Wiley & Sons, 2013.

[169] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[170] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.

[171] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. No. 55, Courier Corporation, 1964.

[172] C. M. Bishop *et al.*, *Pattern recognition and machine learning*, vol. 4. springer New York, 2006.

[173] J. Wishart, "The generalised product moment distribution in samples from a normal multivariate population," *Biometrika*, pp. 32–52, 1928.

[174] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[175] A. Iserles, "Lie groups and the computation of invariants," *University of Cambridge, Department of Applied Mathematics and Theoretical Physics*, 1998.

[176] S. Vijayakumar, A. D'Souza, and S. Schaal, "Lwpr: A scalable method for incremental online learning in high dimensions," 2005.

[177] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: improving classification performance when training data is skewed," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1–4. 00014.

[178] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Prognostics and Health Management, 2008. PHM 2008. International Conference on*, pp. 1–9, IEEE, 2008.

[179] A. Saxena and K. G. (2008), "C-MAPSS Data Set", NASA Ames Prognostics Data Repository." `http://ti.arc.nasa.gov/project/prognostic-data-repository`.

[180] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*, pp. 23–37, Springer, 1995.

[181] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases: PKDD 2003*, pp. 107–119, Springer, 2003.

[182] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *arXiv preprint arXiv:1106.1813*, 2011. 01846.

[183] N. Valeyrie, F. Maurelli, P Patron, J. Cartwright, B. Davis, Y. Petillot, "Nessie V Turbo: a new hover and power slide capable torpedo shaped AUV for survey, inspection and intervention," in *AUVSI North America 2010 Conference*, 2010.

[184] G. Fagogenis and D. Lane, "A variational bayes approach for reliable underwater navigation," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 2252–2257, IEEE, 2015.

[185] G. Fagogenis and D. M. Lane, "Online fault detection and model adaptation for underwater vehicles in the case of thruster failures," in *International Conference on Robotics and Automation, 2016. ICRA 2016.*, IEEE, 2016.

[186] G. Fagogenis, D. Flynn, and D. Lane, "Novel RUL prediction of assets based on the integration of auto-regressive models and an rusboost classifier," in *Prognostics and Health Management (PHM), 2014 IEEE Conference on*, pp. 1–6, IEEE, 2014.

[187] H. Wold, "Partial least squares," *Encyclopedia of statistical sciences*, 1985.

[188] S. Schaal and C. G. Atkeson, "Assessing the quality of learned local models," *Advances in neural information processing systems*, pp. 160–160, 1994.